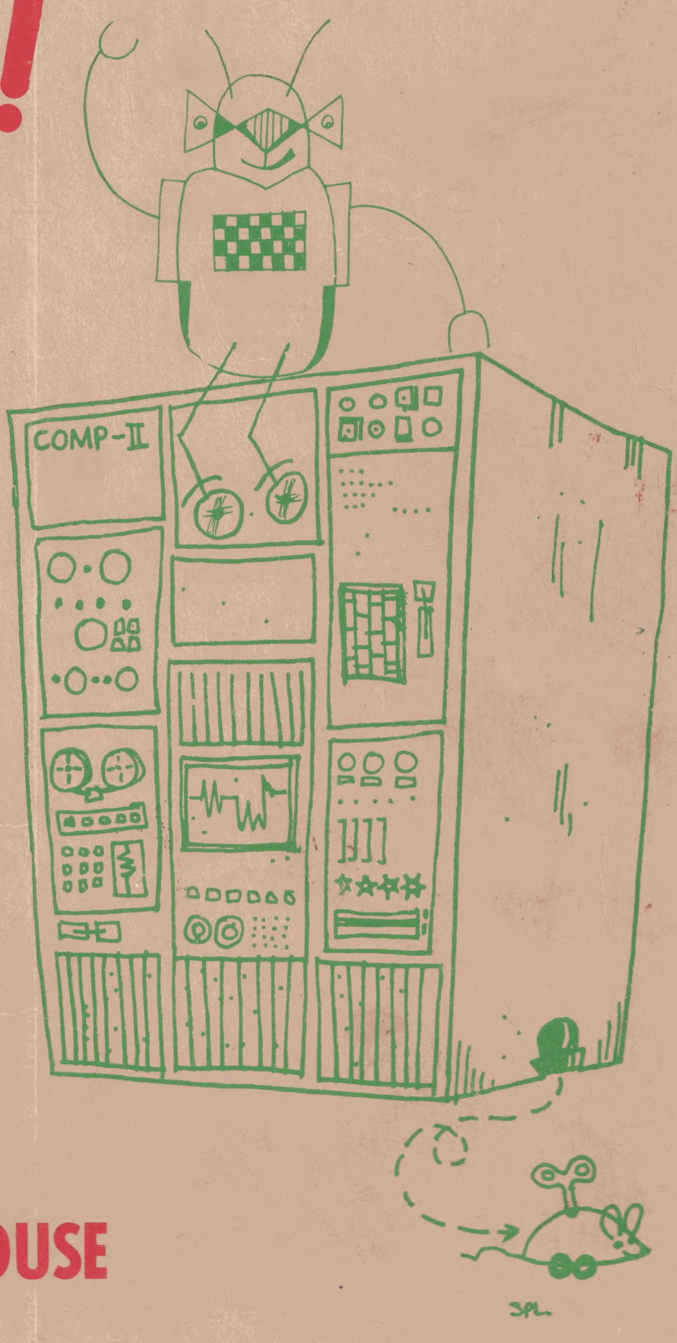
PROGRAMMING IS FUN!



A Step by Step Approach
to Programming
in BASIC for
Secondary Students

by

OLIVER AND NEWHOUSE

PROGRAMMING IS FUN !

A Step by Step Approach to
Programming in BASIC for Secondary
Students

Ron Oliver, B.Sc., B.Ed., Grad.Dip.Comp.
Paul Newhouse, B.Sc., Dip.Ed., Grad.Dip.Comp.
Graphics by Paul Loney

Published Feb. 1982
Reprinted Nov. 1982
Reprinted Aug. 1983

Napier Publications Pty. Ltd.
19 Rayment St., Victoria Park, Western Australia, 6100.
Telephone: 361 3154
©

Printed by Action Press
28 Rudloc Road, Morley
Telephone 276 5442

COMMENTS AND NOTES FOR TEACHERS ON THE USE OF THIS BOOK

The authors of this book have taught computing to secondary school students for a number of years and as such this book is based on their teaching strategies (however it could quite easily fit other strategies). The book is designed to suit a two year course of between one and two hours a week for students between the ages of 13 and 17. All program examples are standard BASIC although they could be converted to any other type of BASIC language.

The text given in each section is of necessity brief and should be supplemented by fuller texts possessed by the teacher. Each section represents one or two lessons of work for the students and should be preceded by a short explanatory teacher talk. It has been assumed that students have access to computers or at least computer terminals. If this is not the case (e.g. you are using cards) minor modifications are suggested throughout the book.

The authors intended this to be largely an examples and exercises book as they believe a 'hands on' approach to computing is the only way to enjoy learning about how to program computers. All sets of exercises are graded so that all students can attempt the first few questions and keener students will find the last few questions challenging. It is also suggested that students work in pairs to help each other and make the most use of hardware available. Each student should try at least one question from each section on a computer.

The material has been tried with classes of 20 students sharing two terminals very successfully. Working in pairs each student gets between 15 and 20 minutes terminal time a week (although more is preferable) with the balance of the time being spent doing the exercises (on paper).

Any constructive criticism is invited at any stage, however any teacher is sure to find a very useful student work book in this text.

The authors wish to thank the staff and students at Willetton Senior High School for their assistance and co-operation in the production of this book.

Note: Several symbols in this book do not correspond exactly to characters on computer keyboards due to typesetting difficulties.

(i) the * should occur in the middle of the line instead of above centre (as here)

(ii) the ‡ should be replaced with #

CONTENTS

PAGE

CHAPTER 1 – GETTING STARTED

1. 1 PRINT STATEMENT 4
1. 2 RUNNING A PROGRAM 6
1. 3 PERFORMING CALCULATIONS 8
1. 4 EXPONENTIAL NOTATION 10
1. 5 LET STATEMENT (1) – NUMERIC STRINGS 12
1. 6 LET STATEMENT (2) – LITERAL STRINGS 14
1. 7 ORDER OF OPERATIONS 16
1. 8 BASIC NOTATION 18
1. 9 REVIEW EXERCISES (1) 20
1.10 REVIEW EXERCISES (2) 21

CHAPTER 2 – MORE COMPLEX STATEMENTS

2.1 INPUT STATEMENT 22
2.2 GO-TO STATEMENT 24
2.3 IF-THEN STATEMENT (1) – GENERAL FORM 25
2.4 IF-THEN STATEMENT (2) – SPECIAL USES 27
2.5 FOR-NEXT STATEMENT (1) – GENERAL FORM 28
2.6 FOR-NEXT STATEMENT (2) – STEPS AND NESTING 30
2.7 SQR FUNCTION 32
2.8 REVIEW EXERCISES (3) 33
2.9 REVIEW EXERCISES (4) 34

CHAPTER 3 – PROGRAM STRUCTURE – GOOD PROGRAMMING

3.1 READABLE PROGRAMS – REM STATEMENT 36
3.2 FLOWCHARTS 38
3.3 TAB STATEMENT 41
3.4 GOSUB STATEMENT 43
3.5 PROJECT (1) 45

CHAPTER 4 – SIMULATION – PROGRAMMING APPLICATIONS

4.1 INT FUNCTION 46
4.2 RND FUNCTION 48
4.3 COUNTING AND SUMMING 49
4.4 SIMULATION 50
4.5 REVIEW EXERCISES 51
4.6 PROJECT (2) 52

CHAPTER 5 – DATA PROCESSING – ANOTHER APPLICATION

5.1 CREATING ARRAYS – DIM STATEMENT 53
5.2 USING ARRAYS 56
5.3 READ AND DATA STATEMENTS 58
5.4 2 DIMENSIONAL ARRAYS 61
5.5 MANIPULATING ARRAYS (1) – LARGEST/SMALLEST ELEMENTS . . 65
5.6 MANIPULATING ARRAYS (2) – SORTING 66
5.7 PROJECT (3) 67
5.8 PROJECT (4) 68

CHAPTER 6 – FILES – DATA STORAGE

6.[illegible] SEQUENTIAL FILES – INTRODUCTION 69
6.2 CREATING AND USING SEQUENTIAL FILES 70
6.3 PROJECT (5) 72

CHAPTER 7 – APPENDIX – ADDITIONAL FEATURES

7.1 INBUILT FUNCTIONS 73
7.2 DEF STATEMENT 74
7.3 ON GO TO STATEMENT 76

SOLUTIONS 78

Chapter 1—GETTING STARTED

Section 1.1 : The PRINT Statement

A program written in the programming language BASIC consists of separate, numbered lines of code with special instructions to the computer, called STATEMENTS.

All programs must finish with the statement END. We will deal with the PRINT statement in this section.

To print messages in BASIC, the message must be enclosed within quote marks, however to print numbers, quote marks are not needed.

To demonstrate this, consider the following programs:

Example 1

```
10 PRINT "A SAMPLE PROGRAM"
20 END
```

This program produces

```
A SAMPLE PROGRAM
```

Example 2

```
10 PRINT 605
20 END
```

This program produces

```
605
```

Note: It is possible to print numbers also within quote marks.

To print different messages on different lines, one uses separate PRINT statements.

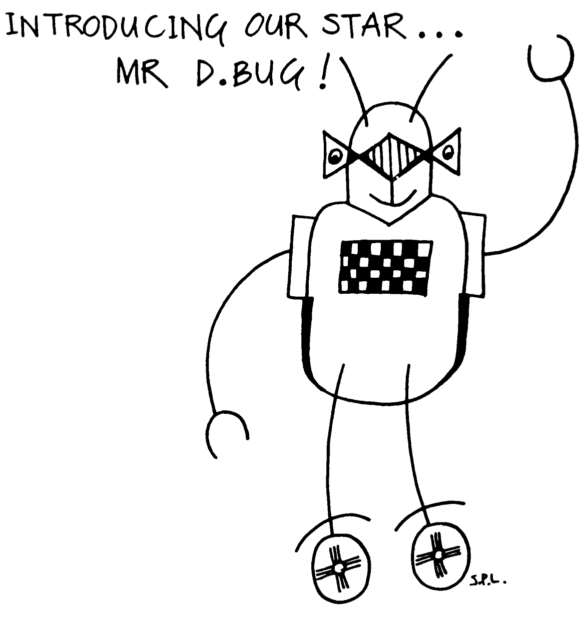
To print distinct messages on the same line, one uses one PRINT statement and the messages separated by commas.

Example 3

```
10 PRINT "HELLO", "500"
20 PRINT "GOOD", "BYE"
30 PRINT 2001
```

will produce

```
HELLO 500
GOOD  BYE
2001
```


INTRODUCING OUR STAR...
MR D.BUG!
S.P.L.

EXERCISE 1.1

1. Show the output from the following programs:

(i)
```
10 PRINT "THE FIRST SECTION"
20 END
```

(ii)
```
10 PRINT "THE", "FIRST", "SECTION"
20 END
```

(iii)
```
10 PRINT "THE"
20 PRINT "FIRST"
30 PRINT "SECTION"
40 END
```

(iv)
```
10 PRINT 1, 2, "THREE"
20 END
```

(v)
```
10 PRINT "1", "2", "THREE"
20 END
```

(vi)
```
10 PRINT 100, 200
20 PRINT 300
30 END
```

2. Each of the following programs has an intentional mistake. Find the line with the mistake and rewrite the whole line correctly.

(i)
```
10 PRNIT "ONE HUNDRED"
20 END
```

(ii)
```
10 PRINT "ONE HUNDRED
20 END
```

(iii)
```
10 PRINT "ONE", HUNDRED
20 END
```

(iv)
```
10 PRINT "ONE" 100
20 END
```

(v)
```
10 PRINT "ONE"
20 PRINT "2"
30 "END"
```

3. Write a program to print COMPUTING IS EASY

4. Write a program to print

```
COMPUTING
IS
EASY
```

5. Write a program to print

```
10 9 8 7 6 5 4 3 2 1
BLAST OFF!
```

Section 1.2 : RUNNING A PROGRAM

(i) NEW

To execute any program on the computer, it is first necessary to enter the program using either cards or typing at the keyboard.
To tell the computer that you wish to enter a program requires the use of the command NEW. The general use of this command is for the user to select a name for the new program and to type

```
NEW name
```

Different computers have different requirements and it is important to check with your teacher to find out the requirements of your computer when using NEW for the first time. Once the command NEW has been accepted by the computer, the current program in main memory is erased and the user simply types his new program.
If cards are being used, the entry of the program cards is automatic.

(ii) RUN

Once the program has been entered into the computer the command RUN is used to cause the computer to execute the program.

```
RUN
```

When the computer receives this command, it executes the program currently stored in main memory following the instructions of the program in the order of the line numbers.

(iii) LIST

To get the computer to type out the program which has been entered with all line numbers in correct sequence, the user must type

```
LIST
```

When this command is received by the computer, the program currently stored in main memory is displayed.

(iv) SAVE

It is the normal procedure to want to store programs in the computer's auxiliary memory and this is achieved by typing

```
SAVE
```

When this command is received by the computer, the program currently stored in main memory is copied to an auxiliary memory and can be reused later.

(iv) OLD

To use a program which has previously been SAVED, requires the use of the OLD command. Because there are normally a number of different programs which have been saved, when using OLD, you must specify the program required

```
OLD AREA
```

When the computer receives this command, it transfers program AREA to main memory and it can then be run or listed as described previously.

(vi) UNSAVE

To erase a program which has been previously saved, one uses the UNSAVE command.

```
UNSAVE AREA
```

Upon receipt of this command, the computer would remove the program AREA from auxiliary storage.

NOTE: Some microcomputers use different commands – check with your teacher if you are using a microcomputer.

EXERCISE 1.2

1. Briefly describe the actions taken by the computer in response to the following commands.
   (i) NEW AREA 1
   (ii) LIST
   (iii) RUN
   (iv) SAVE
   (v) UNSAVE AREA 1
   (vi) OLD AREA 2
   (assume AREA 2 has been previously saved)

2. Give the commands used to cause the following:
   (i) display the program currently in main memory.
   (ii) execute the program currently in main memory
   (iii) transfer a copy of the program currently in main memory to auxiliary memory.
   (iv) erase the program called VOL 1 stored in auxiliary memory
   (v) transfer the program VOL 2 from auxiliary memory to main memory
   (vi) erase the current program stored in main memory

3. (i) Write a program to print your name.
   (ii) List the commands which would be used to enter, list and run this program.

4. Write a program to print the name of your school.

5. Write a program to print the numbers 1, 2, 3, 4 and 5 on separate lines.

6. Write a program to print your name and address as they would appear on an addressed envelope.

Section 1.3 : PERFORMING CALCULATIONS

In BASIC the arithmetic operations are

+	addition
—	subtraction
*	multiplication
/	division
↑	to the power of

To perform calculations in a program one simply uses the PRINT statement followed by the calculation.

Example 4

```
10  PRINT  6 * 5
20  PRINT 120/12
30  PRINT 4 ↑ 2
40  PRINT 5 + 7, 12 – 4
50  END
```

will produce the following output when RUN

```
30
10
16
12      8
```

Example 5

```
10  PRINT 6 * 5
20  PRINT "6 * 5"
30  END
```

will produce

```
30
6 * 5
```

Can you see why?

YES WE DO NEED SOMEONE WITH EXPERIENCE OF V.D.U.'S – BUT THE FACT THAT YOU WATCH A LOT OF TELEVISION IS HARDLY SUFFICIENT...

EXERCISE 1.3

1. What is the output from the following programs?

(i)
```
10  PRINT 8 * 7
20  PRINT 6 + 5
30  END
```

(ii)
```
10  PRINT 4 + 5, 6 * 10
20  PRINT "STOP"
30  END
```

(iii)
```
10  PRINT 120/12
20  PRINT 20 + 10
30  PRINT 200 – 180
40  END
```

(iv)
```
10  PRINT "THIS"
20  PRINT "IS"
30  PRINT "EASY"
40  END
```

(v)
```
10  PRINT 15/3, 50/5
20  PRINT 40 + 5 + 10, 30/6/5
30  PRINT "THE", "END"
40  END
```

2. Find the mistake in the following programs and rewrite the whole line correctly.

(i)
```
10  PRINT 5 * 6
20  PRINT 7 PLUS 5
30  END
```

(ii)
```
10  PRINT "STOP"
20  PRINT "AND"
30  PRINT LOOK
40  END
```

(iii)
```
10  PRINT 8 * 9  7 * 5
20  PRINT 5 + 11  6 + 9
30  END
```

(iv)
```
10  PRINT 45/3
20  PRINT 110 + 120
30  PRINT 60 * 2 : 50 * 2
40  END
```

3. Write a program to add 12 and 25.

4. Write a program to multiply 35 and 10.

5. Write a program to divide 120 by 3.

6. Write a program to subtract 25 from 75.

7. Write a program to calculate 20^3.

8. Write a program to find the average of 15.6, 7.52, 3.90 and 6.57.

Section 1.4 : EXPONENTIAL NOTATION

Most computers using BASIC will only print out numbers with up to 8 digits.

A number like 642 245 will be printed as it stands but 642 245 500 has too many digits and will be printed in exponential form

0.642 2455 E + 9

Similarly 0.0006422455 will be printed

0.642 2455 E − 3

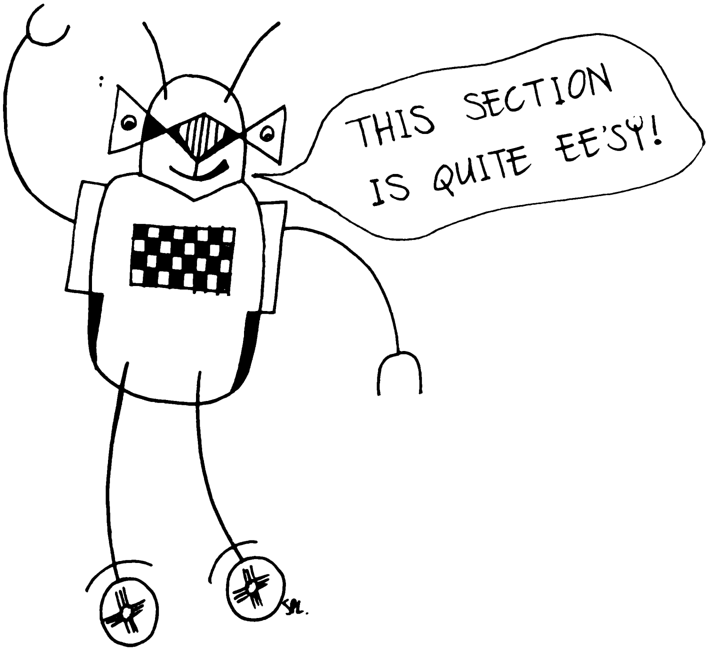
The number following the E indicates how many positions to move the decimal point; + moves it to the right making the number larger; − moves it to the left making the number smaller.

Example 6

(i) PRINT .6E + 2 will output 60

(ii) PRINT 6.4E − 2 will output 0.064

(iii) PRINT .48E + 3 will output 480


THIS SECTION IS QUITE EE'SY!

EXERCISE 1.4

1. Convert these exponential numbers to decimal form.

   (i) 0.62E + 2

   (ii) 4.5964E + 3

   (iii) 5.7E − 2

   (iv) 6.2932E + 5

   (v) 0.72254E − 3

   (vi) 0.2487E + 6

   (vii) 1.00005E + 8

   (viii) 1.75E − 4

   (ix) 7.52E − 3

   (x) 2.4557E + 6

2. Write in exponential notation

   (i) 0.000056

   (ii) 750000

   (iii) 100000000

   (iv) 0.000001

   (v) 567534500

   (vi) 56.230

   (vii) 0.00005678

   (viii) 3010

   (ix) 256000

   (x) 2580.01

3. Write in decimal form and simplify

   (i) 5E + 2 x 4E + 3

   (ii) 2E − 2 x 5E + 4

   (iii) 6.4E + 2 ÷ 8E + 1

   (iv) 5.6E − 3 ÷ 7E − 4

   (v) 2.8E + 10 + 1.2E + 9

4. Write a program to calculate 1000^4.

5. Write a program to add 0.6E + 10 and 0.45E + 9.

6. Write a program to evaluate 9 ÷ 5E + 10.

7. Write a program to evaluate 0.64E + 5 x 0.5E − 2.

8. Write a program to calculate 0.56E + 8 ÷ 0.7E + 5.

Section 1.5 : THE LET STATEMENT (1) – NUMERIC STRINGS

The LET statement is used to store numbers and messages in BASIC.

Consider the program:

```
10  LET A = 5
20  LET B = 6
30  LET C = A * B
40  END
```

This causes the computer to store the numbers in memory spaces represented as follows:

A	5
B	6
C	30

Using PRINT, one can examine the numbers stored.

Example 7

```
10  LET A = 5
20  LET B = 6
30  LET C = A * B
40  PRINT A, B, C
50  END
```

will output

```
5    6    30
```

The numbers can be stored in memory spaces

A, A1, A2, ... A8, A9, A0.
B, B1, B2, ... B8, B9, B0.
Z, Z1, Z2, ... Z8, Z9, Z0.

EXERCISE 1.5

1. Find the mistake in each of the following programs and rewrite the incorrect lines of code.

(i)
```
10 LET A = 4.5
20 LET B = 2
30 LET 5 = C
40 PRINT A + B + C
50 END
```

(ii)
```
10 LET D = 42
20 LET E = 7
30 LET F = D ÷ E
40 PRINT D, E, F
50 END
```

(iii)
```
10 LET G = 5
20 LET H = 30
30 LET I = 40
40 LET K = G + H + I
50 PRINT G H I K
60 END
```

2. Show the output from each of the following programs:

(i)
```
10 LET F = 50
20 LET G = 5
30 LET H = F * G
40 PRINT F, G, H
50 END
```

(ii)
```
10 LET K = 5
20 LET J = 2
30 LET I = K + J
40 PRINT I
50 END
```

(iii)
```
10 LET M = 100.0
20 LET N = 15.5
30 LET O = M – N
40 PRINT M, N
50 END
```

3. Write a program to add 15, 16 and 25.

4. Write a program to evaluate a^2 when a = 6 and a second time when a = 7.

5. Write a program to add 25 and 15, to also multiply 25 and 15.

6. Write a program to multiply 7, 10 and 6.

7. Write a program to evaluate 6ab + 3bc when a = 5, b = 10 and c = 6.

Section 1.6 : THE LET STATEMENT (2) – LITERAL STRINGS

In order to store words in BASIC, one uses memory spaces labelled

A$, B$, C$ .... Z$
A1$, B1$, C1$ .... Z1$ etc.

Consider the program segment:

```
10  LET A$ = "BILL"
20  LET B$ = "SMITH"
```

This causes the computer to store the words in memory spaces represented as follows:

A$	BILL
B$	SMITH

Using PRINT, one can examine the contents of the memory spaces.

Example 8

```
10  LET A$ = "BILL"
20  LET B$ = "SMITH"
30  PRINT A$, B$
40  END
```

will output

```
BILL SMITH
```

HALLO, B.J. ?
THE COMPUTER'S
DOWN AGAIN...
S.P.L.

EXERCISE 1.6

1. Find the mistake in each of the following programs and rewrite the line correctly.

(i)
```
10  LET A$ = "JOHN"
20  LET B$ = SMITH
30  PRINT A$
40  PRINT B$
50  END
```

(ii)
```
10  LET C$ = "HELLO"
20  LET D = "THERE !!"
30  PRINT C$, D
40 END
```

(iii)
```
10  LET A = 5
20  LET B$ = "TIMES"
30  LET B = 10
40  LET A$ = "IS"
50  PRINT A, B$, B A$, A * B
60  END
```

2. Show the output from each of the following programs.

(i)
```
10  LET D$ = "SAMPLE"
20  LET E$ = "PROGRAM"
30  PRINT D$, E$
40  END
```

(ii)
```
10  LET M$ = "PLUS"
20  LET N$ = "="
30  LET B = 6.5
40  LET C = 3.5
50  LET D = B + C
60  PRINT B, M$, C, N$, D
70  END
```

(iii)
```
10  LET A$ = "EASY"
20  LET B$ = "PROGRAMMING"
30  LET C$ = "IS"
40  LET D$ = "?"
50  PRINT C$, B$, A$, D$
60  PRINT B$, C$, A$
70  END
```

3. Write a program to store your name, address and postcode in A$, B$ and C$ and to print out these details.

4. Write a program to store your name, class and school and to print out these details.

5. Write a program to store the subjects you study at school and to print them out.

6. Write a program to store the words HOW, YOU, DO and to print the message HOW DO YOU DO.

Section 1.7 : ORDER OF OPERATIONS

When the computer evaluates expressions in BASIC it uses the arithmetic operations in the following order:

1. Brackets
2. Indices
3. Multiplication and Division (left to right)
4. Addition and Subtraction (left to right)

When the same operations are met in one expression, the computer uses them in order from left to right.

Example 9

(i) Evaluate 5 + 2 * 3

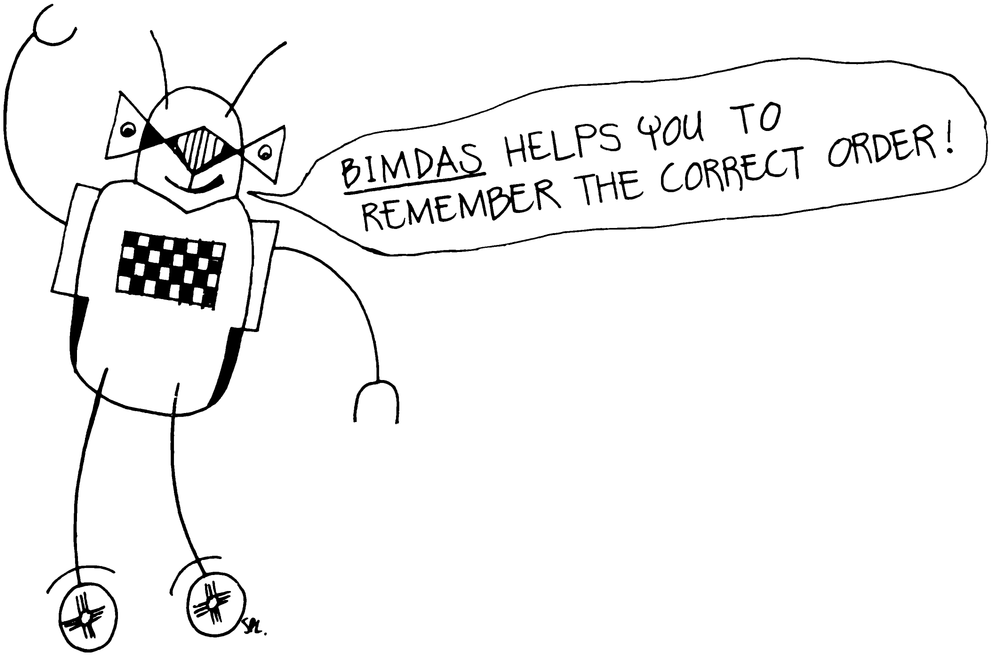
Answer: 5 + 6 = 11

(ii) Evaluate 10 – 2 + 3 ↑ 2

Answer: 10 – 2 + 9 = 8 + 9 = 17

(iii) Evaluate (5 + 3) – (4 + 3)/14

Answer: 8 – 7/14 = 8 – ½ = 7.5


BIMDAS HELPS YOU TO REMEMBER THE CORRECT ORDER!

EXERCISE 1.7

1. Evaluate the following:

   (i) 5 + 2 ↑ 2

   (ii) 3 + 2 * 3 – 1

   (iii) 25 – 5 + 2 * 3

   (iv) 6 * 2 ↑ 2 + 5

   (v) 100/4 + 6

   (vi) 75 + 10/2

   (vii) 110 + 2 * 8

   (viii) 300 + 50/10 – 5

   (ix) –100 + 5 ↑ 2 + 1

   (x) 25 * 2 * 5/10

2. Evaluate the following:

   (i) 200 + 5 * 2 – 5

   (ii) 25 – 5 + 3 * 2

   (iii) 10 ↑ 2 + 3 * 5

   (iv) 20 * (2 + 5)/5

   (v) (60 * 5 * 2) ↑ 2

   (vi) (5 + 2) * (3 + 5) * 10/2

   (vii) (1000 – 200 * 3) ↑ (11 – 13 + 4)

   (viii) 2 * 3/5 + 4/5

   (ix) 750 – 25 * 10 – 200 * 3

   (x) (23 + 77) ↑ 4 * 2 + 1

3. Using A = 12, B = 6, C = 3 and D = 2, evaluate:

   (i) A + B/C

   (ii) A * D/B

   (iii) B/C ↑ D

   (iv) (A + B + D)/D

   (v) B ↑ C/B ↑ D

   (vi) C * B + A ↑ D

   (vii) (A/B/D) ↑ B

   (viii) B/C/D * A

   (ix) B/(C + D) ↑ D

   (x) A + B + C + D/A + B + C + D

4. Write a program to evaluate 10 * 5 + 4/2.

5. Write a program to evaluate 5 ↑ 2/5 + 6.

6. Write a program to evaluate (6 + 12)/3 – 5 + 4.

7. Write a program to evaluate A + B * C – D when A = 5, B = 10, C = 3 and D = 4.

8. Write a program to evaluate A * B/C + D – E when A = 5, B = 30, C = 10, D = 6 and E = 15.

9. Write a program to evaluate D/(C + E) * F when D = 40, C = 5, E = 3 and F = 2.

Section 1.8 : BASIC NOTATION

When writing expressions in BASIC, one must be careful to include all arithmetic operations which are often omitted in algebra.

Example 10

(i) Express $\frac{1}{2}a + 5t^2$ in BASIC

Answer: `1/2 * A + 5 * T ↑ 2`

(ii) Express $\frac{a^2}{b - c}$ in BASIC

Answer: `A ↑ 2/(B – C)`

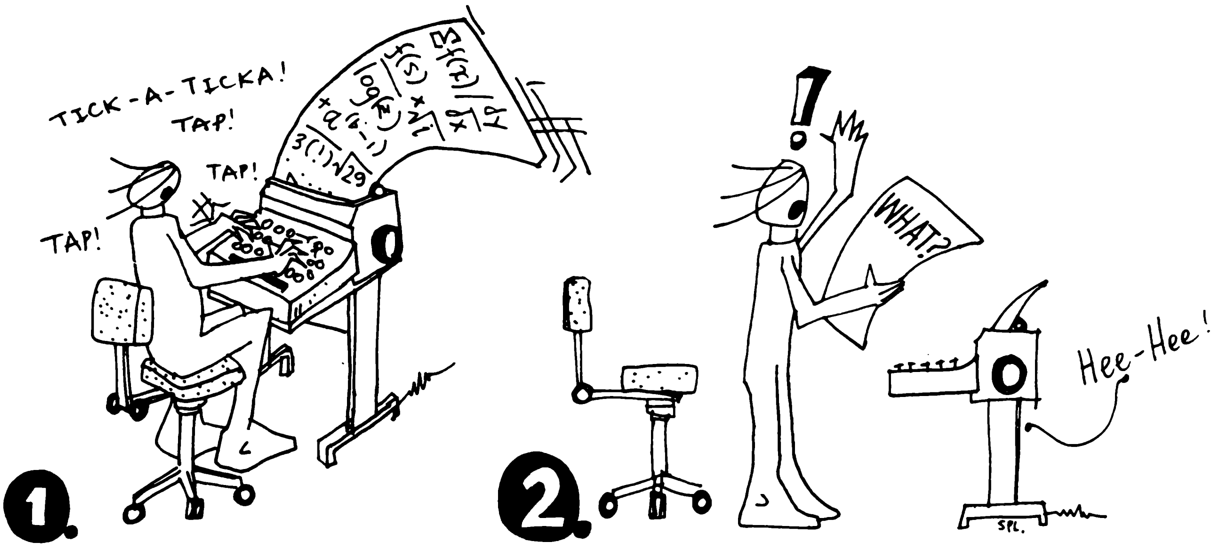
Programs can be written to evaluate expressions as in the following example:

Example 11

Write a program to evaluate 2abc when $a = 5$, $b = 4$ and $c = 3$

Answer:

```
10  LET A = 5
20  LET B = 4
30  LET C = 3
40  LET D = 2 * A * B * C
50  PRINT D
60  END
```


TICK-A-TICKA!
TAP!
TAP!
TAP!
1.
!
WHAT?
Hee-Hee!
2.
SPL.

EXERCISE 1.8

1. Express the following in BASIC notation.

   (i) $5a + b$

   (ii) $2\ prt$

   (iii) xy^2

   (iv) $\frac{3}{4}\ ab$

   (v) $2\ (a + b)$

   (vi) $\frac{1}{a} + b$

   (vii) $\frac{1}{3} l^2 b$

   (viii) $(a + b)^3$

   (ix) $2rs + 3r^3$

   (x) $\frac{1}{2}\ (a^2 + b^3)$

2. Express the following in BASIC notation.

   (i) $2pr + 3st$

   (ii) $u^2 + 2as$

   (iii) $\frac{1}{a + b}$

   (iv) $\frac{b + c}{2a}$

   (v) $ut + \frac{1}{2} at^2$

3. Using A = 12, B = 6, C = 3 and D = 2, express the following in BASIC notation and evaluate the expression.

   (i) $\frac{1}{2}\ bc^2 + ad$

   (ii) $\frac{ac}{b - d}$

   (iii) $a^d + cd^d$

   (iv) $a^2 + b^2 + ab$

   (v) $\left(\frac{a - b + c}{c}\right)^2$

4. Write a program to evaluate $2ab + 3cd$ when $a = 5$, $b = 3$, $c = 4$ and $d = 10$.

5. Write a program to evaluate $(4a - 2\ bc)^2$ when $a = 5$, $b = 3$ and $c = 1.5$.

6. Write a program to evaluate $u^2 + 2as$ when $u = 5.5$, $a = 3.7$ and $s = 4.2$.

7. The circumference of a circle is calculated using the formula $c = 2\pi r$.
   Write a program to find the circumference of a circle with radius 6 cm if π is 3.14.

8. The perimeter of a rectangle is found using the formula $P = 2\ (l + w)$.
   Write a program to find the perimeter of a rectangle with sides length 144m and width 86m.

Section 1.9 : REVIEW EXERCISES (1)

1. Evaluate the following expressions, show the steps involved.

   (i) 5 – 2 + 6

   (ii) 3 * 2 + 4

   (iii) 35 – 2 * 3

   (iv) 2 + 3/4

   (v) 24/2 * 3

   (vi) 1/2 + 3

   (vii) 4 + 2 ↑ 2

   (viii) 7 * 3 + 2

   (ix) 35 – 6 * 2 ↑ 3

   (x) 21 + 4/2 + 3

2. Using A = 4, B = 3, C = 2 and D = 5, evaluate

   (i) A + B + C/A

   (ii) A * B/C

   (iii) (B + C + D)/(D * C)

   (iv) A * B * C – A ↑ C

   (v) ((B + C)/D) * B

3. Rewrite the following equations in BASIC notation.

   (i) $s = ut + \frac{1}{2} at^2$

   (ii) $p = \frac{1}{2} ab^3 - 3d$

   (iii) $v = (u^2 + 2as)^{1/2}$

   (iv) $d = \dfrac{x - y}{x^2 + y^2}$

   (v) $e = \dfrac{xy + yz}{2xy}$

4. Using the formula $s = ut + \frac{1}{2} at^2$, write a program to evaluate and print s, given u = 10.54, t = 5.63 and a = 2.54.

5. Write a program which takes three numbers, A = 50.6, B = 74.1 and C = 5.62 and outputs the average of these.

6. What would be the output from the following segment?

```
10  LET A = 6
20  LET B = 3
30  PRINT B, A, A * B
```

Section 1.10 : REVIEW EXERCISES (2)

1. Find the errors in the following BASIC program lines. Write out the lines correctly formed.

```
10  LET A = 10
20  LET AB = 20
30  C = A * AB
40  PRINT C
50  PRINT END
```

2. What would be the result of

```
10  PRINT "THIS IS IT"?
```

3. What would be output from this program segment?

```
10  LET A = 5
20  PRINT "A =" A
```

4. If A = 5, B = 10 and C = 2, evaluate the following:

   (i) A + B – 2

   (ii) B/A * C

   (iii) A + B * C + A

   (iv) B * C ↑ 2

5. Write the following in BASIC notation:

   (i) $4x \div 3$

   (ii) $8y + 7$

   (iii) $x^2 + y^2$

   (iv) $(a + b)^{1/2}$

   (v) $(4ab)\ (5bc)$

   (vi) $(r + 4)\ (r - 3)$

   (vii) $a + (b + c) \div 2$

   (viii) $2x + 2y \div (4r + 5t)$

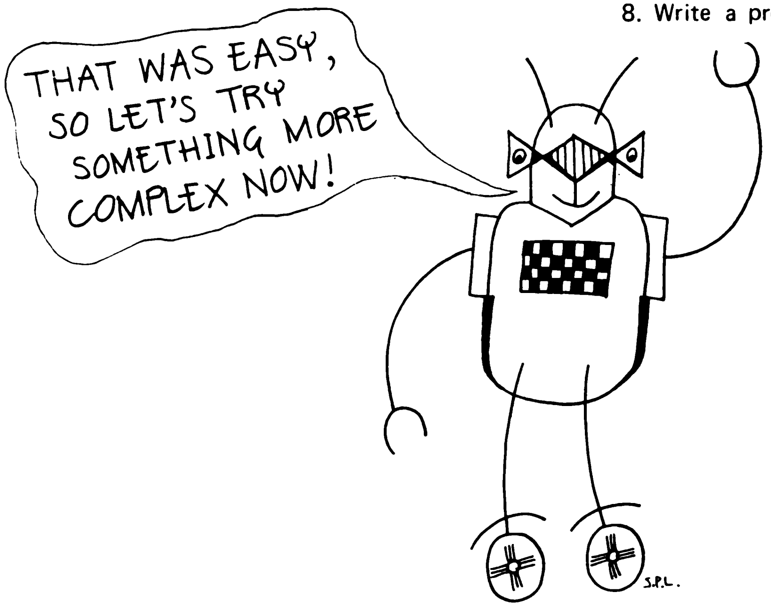
   (ix) $x^2 + 3x + 4$

   (x) $6p \div 5n^2$

6. Write a program to find the value of y when $x = \frac{11}{4}$ given $y = 3x^3 - 5x^2 + 2x - 17$.

7. Write a program to find the average of 15.7, 30.42, 160.3, 49.7, 16.4 and 1.79.

8. Write a program to find the 4th power of 16.7.


THAT WAS EASY, SO LET'S TRY SOMETHING MORE COMPLEX NOW!
S.P.L.

Chapter 2—MORE COMPLEX STATEMENTS

Section 2.1 : THE INPUT STATEMENT

The INPUT statement is used in BASIC to enable programs to use varying data on separate executions.

Consider the piece of code

```
10  PRINT "TYPE A NUMBER"
20  INPUT A
30  PRINT "TYPE A WORD"
40  INPUT A$
```

When this program runs, the screen shows

```
TYPE A NUMBER
?
```

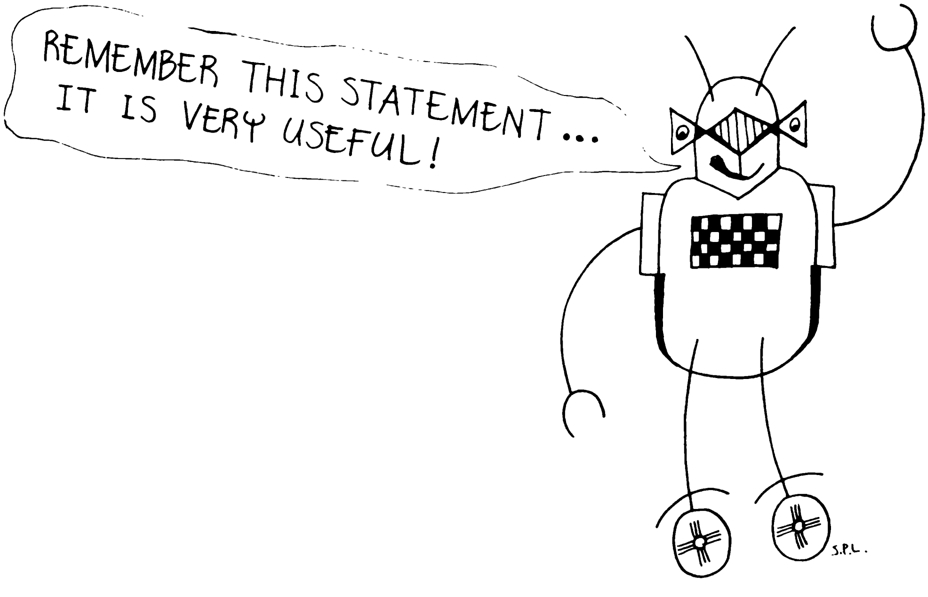
The computer then waits until a number is typed on the keyboard and this number is then stored in A.

Similarly, the computer waits for input at line 40 and stores whatever is typed in A$.

Note:

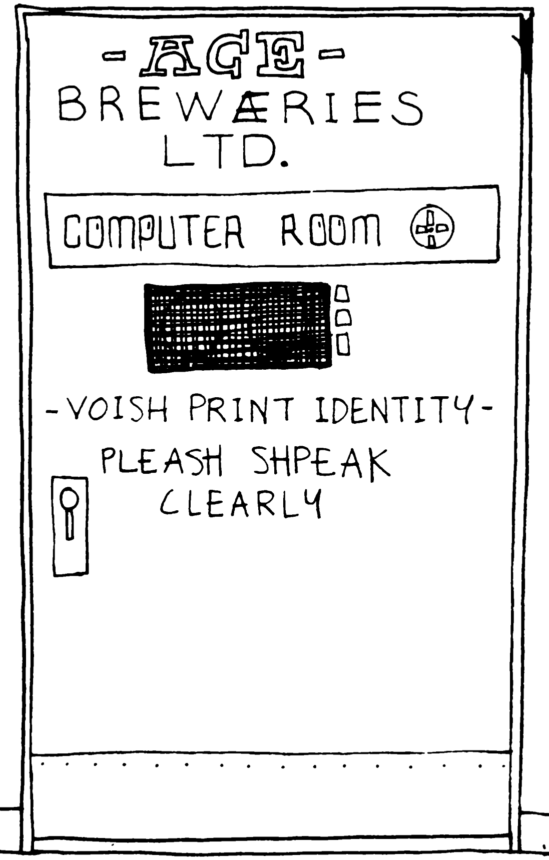
If the user attempts to store a word in A, i.e. a word is typed at line 20, an error message will result.

Numbers, however, can be stored in A$ at line 40.


REMEMBER THIS STATEMENT... IT IS VERY USEFUL!
S.P.L.

EXERCISE 2.1

1. Write a program to input any number and to output that number doubled.
2. Write a program to input two numbers and to output the difference of the two numbers.
3. Write a program to input three numbers and to output the sum of these numbers.
4. Write a program to input 2 numbers and to output the product of these numbers.
5. Write a program to change the temperature in fahrenheit to celsius where C = 5 (F − 32)/9.
6. Write a program to calculate the perimeter of any rectangle. Use INPUT for the length and width.
7. Write a program to find the area of a circle given the radius.
8. Write a program to input 3 numbers, called U, A and T and to calculate V given the formula V = U + AT.
9. Write a program to input any 5 numbers and to find the average of those numbers. (INPUT A, B, C is allowed)
10. Write a program to input your name, address and postcode and to output them in reverse order.
11. Write a program to input the model of a car, its colour and year of production. Your program must output the model, colour and age of the car, on one line.
12. Write a program to input your name, school and year level and to print these out.
13. Write a program to input a person's name and age which will tell the user the year he was born in.



Section 2.2 : GOTO STATEMENT

The value of a computer is that it can rapidly repeat a segment of program code as many times as desired. The GOTO statement is a command which helps to allow this to happen. It is an unconditional branch to another part of the program code.

e.g.

```
45 GOTO 10
```

– this instructs the computer to branch back to statement 10.

Note: GOTO's can be used to branch either forward or backward, however there are often better ways to branch. So you should minimise the use of GOTO.

Example 12

```
10 LET P = 6
20 GOTO 40
30 PRINT P
35 GOTO 60
40 PRINT "THE NUMBER IS     " ;
50 GOTO 30
60 END
```

The output is : THE NUMBER IS 6

The lines are executed in the following order : 10 20 40 50 30 35 60.

EXERCISE 2.2

In each case write down the output on a run and write down the order in which the lines would be executed.

1.
```
 5 GOTO 15
10 PRINT "THIS" ;
15 PRINT "SAMPLE"
20 END
```

2.
```
10 PRINT "UN" ;
15 GOTO 40
30 LET A = 10
40 PRINT "CONDITIONAL"
50 PRINT "LOOPS"
60 PRINT A
70 END
```

3.
```
10 LET N = 1
20 PRINT N
30 LET N = N + 2
40 GOTO 10
50 END
```

4.
```
 5 LET A$ = "NAME"
 6 GOTO 40
10 PRINT "WHAT" ,
20 PRINT "IS",
30 GOTO 70
40 PRINT "YOUR",
50 PRINT A$
60 GOTO 10
70 END
```

5.
```
10 S = 0
20 I = 0
30 I = I + 1
40 S = S + I
50 GOTO 30
60 END
```

Section 2.3 : IF – THEN STATEMENT

In the last set of exercises you should have found that the programs for questions 3 and 5 would continue for ever (or until you interrupted the program). Therefore we need a statement to get out of these infinite loops. We often want to stop when a certain condition is true (e.g. say when we have done the loop in question 5, twenty times).

The IF – THEN statement has the general form:

```
nn IF condition THEN mm
```

where n, m – are whole numbers
condition – is something which can be true or false.

If the condition is true it branches to line 'mm' otherwise it continues through to the next program line.

In question 5 we could insert

```
35 IF I = 20 THEN 60
```

and now after adding 1 + 2 + ... + 20 it would go to the end.

The conditions we will use are of the form : numerical or literal string, relational operator, numerical or literal string.

Possible relational operators are:

=	equal to
>	greater than
<	less than
> =	greater than or equal to
< =	less than or equal to
< >	not equal to

Example 13

```
 10 PRINT "INPUT A NUMBER"
 20 INPUT N
 30 IF N > 0 THEN 70
 40 IF N < 0 THEN 90
 50 PRINT N ; "IS NOT POSITIVE NOR NEGATIVE"
 60 GOTO 100
 70 PRINT N ; "IS A POSITIVE NUMBER"
 80 GOTO 100
 90 PRINT N ; "IS A NEGATIVE NUMBER"
100 END
```

EXERCISE 2.3

1. (i) What does the example above do?

   (ii) What would be the output if a 6 was input? −51?, 0?

2. What would be the output from this section of code?

```
10  LET M = 4
20  PRINT M
30  IF M > 100  THEN  60
40  LET M = M + 10
50  GOTO 20
60  END
```

3. What would the output be here?

```
10  PRINT "1"
15  LET P = 1
20  PRINT P + 1
30  PRINT "CAN YOU COUNT?"
40  LET P = P + 1
50  IF P < 10  THEN  20
60  PRINT "NO!!"
70  END
```

4. Write a program which types out 2, 12, 22, 32, ... 102 and then stops.

5. Design a program to accept positive numbers but which rejects any other number by printing an error message.

6. Write a program to find the product of the first 20 counting numbers.

7. Write a program to make a table of the squares of the negative numbers, starting at −1 and stopping at −20.

8. Include a few extra pieces in your program from question 7 so that it includes 'cubes' of the negative numbers in the table.

9. Design a program to allow a user to guess a number you have chosen. Each time tell him whether he is 'too high' or 'too low'. When he has guessed the number, stop and give him a 'pat on the back'. Can you also tell him how many guesses he had at your chosen number?

10. The Fibonacci series of numbers is 1, 1, 2, 3, 5, 8, 13, 21, ... Can you see how the series is constructed? Now write a program to type out the first 25 terms in the series across the page.

IF YOU HAVE FINISHED THIS SECTION SUCCESSFULLY, THEN GO FORWARD ONE PAGE!

Section 2.4 : IF – THEN STATEMENT (2)

Example 14

```
10  PRINT "WHAT IS THE NAME OF THE CAPITAL OF WESTERN AUSTRALIA?"
20 INPUT A$
30 IF A$ = "PERTH" THEN 60
40 PRINT "NO. THE ANSWER IS PERTH."
50 GOTO 70
60 PRINT "CORRECT!! WELL DONE"
70 END
```

Notice that in line 30 a literal string can be used in a condition.

EXERCISE 2.4

1. Write a program to ask one general knowledge question. It should tell the user whether he is correct and if not what the correct answer is.
2. Write a program which multiplies any two numbers together which the user inputs. After giving the answer it should ask the user whether she wants another go, if she types YES it should start the program again.
3. Write a program to divide any two numbers and allow the user to use it as many times as he wants to.
4. Design a program to find the square root of any number. The program must stop and give an error message if a negative number is input. [Hint: Use a statement with LET B = A ↑ 0.5]
5. Write a program which divides any number by a second number but stops and gives an error message when the second number is 0.
6. Design a program which will accept any two numbers and print them out in order of size (magnitude).
7. Write a program to accept any two words and print them out in alphabetical order.

   [Note: The computer reads
   ADAM as < FRED
   and ADAM as < ALAN]
8. Design a program to order any 3 numbers. (Hint: you will need to use a few IF – THEN statements).
9. Write a program to ask 5 general knowledge questions. At the end of the last question the user should be told how many she has managed to get correct.
10. Design a program to calculate bank balances. The user should input the present balance and then the amount deposited (positive number) or withdrawn (negative number). Each run should produce a document of the form:

```
   CALCULATED NEW ACCOUNT

   DEPOSIT        )  $
or WITHDRAWAL     )

   NEW BALANCE       $
```

Section 2.5 : FOR – NEXT STATEMENT (1)

Example 15

What does this program do?

```
10  LET P = 1
20  LET N = 1
30  LET N = N + 1
40  LET P = P * N
50  IF N < 20 THEN 30
60  PRINT "THE PRODUCT IS" P
70  END
```

In this program we know that we want to repeat line 40, 20 times thus instead of the IF statement and counter (line 30) we may use a loop.

Example 16

```
10  LET P = 1
20  FOR N = 1 TO 20
30  LET P = P * N
40  NEXT N
50  PRINT "THE PRODUCT IS" P
60  END
```

The loop [lines 20, 30, 40] is executed 20 times as N has a value starting at 1 and going through to 20.

Here are two more examples using loops:

Example 17

```
10  FOR I = 1 TO 4
20  PRINT "HELLO",
30  NEXT I
40  END
```

The output on a RUN is:

```
HELLO      HELLO      HELLO      HELLO
```

Example 18

```
10  FOR J = 10 TO 14
20  PRINT J,
30  NEXT J
40  END
RUN
10      11      12      13      14
```

EXERCISE 2.5

List the output for each:

1. (i)

```
10  FOR J = 1 TO 6
20  PRINT "GOODBYE"
30  NEXT J
40  END
```

(ii)

```
10  LET C$ = "NAME"
20  FOR N = 1 TO 5
30  PRINT C$ ;
40  NEXT N
50  END
```

(iii)

```
10  LET X = 6
20  FOR M = 1 TO 10
30  PRINT X,
40  PRINT M
50  NEXT M
60  END
```

(iv)

```
10  FOR X = 12 TO 16
20  PRINT "FRED",
30  PRINT "IS DEAD"
40  NEXT X
50  END
```

(v)

```
10  FOR K = 20 TO 31
20  PRINT K + 2
30  NEXT K
40  END
```

Write programs using loops to do the following tasks:

2. Print out the word DOLLARS five times along a line.

3. Print a string of thirty five *'s across a line, each being next to the other.

4. Output the numbers 1 to 20.

5. Modify your program from question four to output all the even numbers up to and including 40.

6. Print the odd numbers from 1 to 21 across the page.

7. Print the whole numbers, their squares and square roots from 1 to 10. Use a heading for each column.

8. Print out the 5 times table.

9. Modify your program from question 8 to allow the user to select the times table he wants printed.

10. Print out the fractions 1, $1\frac{1}{25}$, $1\frac{2}{25}$ .. $1\frac{24}{25}$ along with the decimal value of each.

11. Add the numbers from 1 to 100 and output the result.

AL, HAVE GONE HOME.
SORRY THINGS ARE IN A
BIT OF A MESS. DON'T
WORRY. IN HASTE.
JIM

Section 2.6 : FOR – NEXT STATEMENT (2)

The following program code will perform the task from question 10.

Example 19

```
10  LET S = 0
20  FOR I = 1 TO 100
30    LET S = S + I
40  NEXT I
50  PRINT "THE SUM IS" S
60  END
```

The variable S acts as a 'summing' variable.

More on FOR . . . NEXT . . . loops

All loops are of the form,

```
F   FOR k = n to m STEP p
     .
     .
     .
     .
    NEXT K
```

Where k is an alphabetic letter
m, n, p are expressions with integer values
n and m give the range of values k takes on
p gives how much k changes by on each pass of the FOR statement.

```
So, FOR J = 10 TO I + 6 STEP 2
    (CODE)
    NEXT J
```

If I = 10 the (CODE) will be executed for J = 10. 12, 14, 16 in order.

Note: STEP p is optional, if it is not included it takes on a default value of 1.

EXERCISE 2.6

1. Write down the expected output from this segment of code

```
50  LET C = 20
60  FOR K = 6 TO C STEP 2
70    PRINT K
80  NEXT K
```

2. What is the output in the following code segment?

```
 90  FOR S = 1 TO 10
100    LET A = S * S
110    PRINT S, A
120  NEXT S
```

3. How many times will this loop be executed?

```
100  LET M = 2
110  FOR I = M + 5 TO M * M + 16 STEP 3
120  NEXT I
```

4. Write a program to add the numbers from 40 to 100 and output the result.

5. Modify your program from question 4 to allow the user to add all the even numbers between any two even numbers he wishes.

6. Write a program to print out a table of squares of the numbers between 40 and 55. See if you can put a heading on the table and list the numbers as well as the squares.

7. Write a program to calculate the factorial value of any number.

   e.g. 6 factorial (written 6!) = 6 x 5 x 4 x 3 x 2 x 1

8. Write a program to find the prime numbers between 1 and 100 (use a loop and the INT function p46).

9. What is output by the following piece of code?

```
10  LET T = 0
20  FOR I = 0 TO 2
30      FOR J = 1 TO 20
40        LET T = T + 1/(J + 10 * I)
50      NEXT J
60      PRINT T,
70  NEXT I
80  END
```

Note: 1. A FOR . . . NEXT loop may be used inside another loop [this is called 'nesting' loops].

2. The indentation of the text on lines 30, 40, 50 & 60 helps the readability of the program.

The solution to question 9 is that it outputs the answer to (1 + ½ + 1/3 + ¼ + . . . + 1/20) and then the answer to (1 + ½ + 1/3 + ... 1/30) and then the answer to (1 + ½ + 1/3 + . . . + 1/40), each added to the one before.

10. Use nested FOR – NEXT loops to have the computer print out a multiplication table of the numbers 12 to 15.

11. If a family owes $30,000 on their house for which they are paying 12% interest per year and repayments of $4,000 per year, how much will still be owed on the house after 10 years? 15 years? Use a FOR – NEXT loop in a program to answer this problem.

    (Recall, Interest = 12% of Amount
    = (12/100) * Amount)

    Modify your program to allow the user to input any percentage interest, amount owing and repayment amount.

Section 2.7 : THE SQR FUNCTION

To find the square root of a number, BASIC uses the function SQR.

The correct form is SQR (number).

It is important to only find the square root of non negative numbers or an error message occurs.

Example 20

```
(i)  PRINT  SQR (16)             will output 4.

(ii) 10  LET A = 25
     20  PRINT SQR (A)           will output 5.
```

EXERCISE 2.7

1. What will be the output of the following programs?

```
(i)   10  LET A = 90
      20  LET B = 10
      30  LET C = A/B
      40  PRINTSQR(C) "ISSQUARE ROOT OF"
          C
      50  END

(ii)  10  LET A = SQR(16)
      20  LET B = SQR(100)
      30  PRINT 2 * A + B
      40  END

(iii) 10  LET A = 81
      20  LET B = SQR(A)
      30  LET C = SQR (B)
      40  PRINT A, B, C
      50  END

(iv)  10  LET A = 1/16
      20  LET B = SQR(A)
      30  LET C = 1/B
      40  PRINT C, B, A
      50  END
```

2. Write a program to find the square root of 455.

3. Write a program to input any number and to output the square root of that number.

4. Write a program to input any number and if the number is positive, to find the square root of that number. If the input number is negative the program should print an error message and stop.

5. Write a program to write out a table with the whole numbers from 1–20 with their squares and square roots.
   Your program should print headings also.

6. Write a program which will input the area of a square in square metres and print out the lengths of the sides of the square in metres.

7. Write a program to calculate the length of the hypotenuse h, of a right triangle, given the lengths a, b of the other sides.

Section 2.8 : REVIEW EXERCISES (3)

1. Show the output from each of the following program segments.

(i)

```
10  PRINT "THIS"
15  GO TO 45
20  PRINT "NOT"
25  PRINT "VERY"
30  GO TO 35
35  PRINT "SENSIBLE"
40  STOP
45  PRINT "IS"
50  GO TO 20
```

(ii)

```
10  LET A = 1
15  PRINT A ↑ 2
20  IF A> = 10 THEN 40
30  LET A = A + 3
35  GO TO 15
40  PRINT "STOP"
```

(iii)

```
10  LET A = 7
15  LET B = 5
20  PRINT A * B
30  LET A = A – 2
35  LET B = B – 1
40  IF A > B THEN 20
50  PRINT A, B
```

(iv)

```
10  FOR I = 1 TO 10
20  LET I = I + 2
30  PRINT I
40  NEXT I
```

(v)

```
10  LET A1 = 10
20  LET B1 = 15
30  PRINT B1 – A1
40  LET C1 = B1 + A1
50  IF C1 > 50 THEN 80
60  LET B1 = B1 + 10
70  IF B1 < 35 THEN 30
80  PRINT A1, B1
```

2. Show the output from each of the following program segments.

(i)

```
20  FOR I = 1 TO 5
25  PRINT "I"
30  NEXT I
```

(ii)

```
35  FOR J = 1 TO 30 STEP 5
40  PRINT J + 2,
45  NEXT J
```

(iii)

```
50  FOR K = 5 TO 2 STEP –1
55  PRINT K ↑ 2
60  NEXT K
```

(iv)

```
65  FOR I = 1 TO 3
70  FOR J = 1 TO 2
75  PRINT I + J, I * J
80  NEXT J
85  NEXT I
```

(v)

```
 90  FOR K = 1 TO 10 STEP 16
 95  PRINT K
100  NEXT K
```

3. Write a program to print out the 5 times table.

4. Write a program which allows the user to select the times table he wants printed.

5. Write a program to input any 5 numbers and to print out the sum of these numbers.

6. Write a program to input numbers which will add these numbers and will stop and give the progressive total when 999 is entered.

7. A company pays its employees a travelling allowance for using their own cars. The allowance is 15c per kilometre for cars with engine capacity less than 1600cc and 18c per kilometre for cars with engine capacity over 1600cc.
   Write a program to calculate the petrol allowance.

8. The State Energy Commission charges people for electricity using the following tariff
   6.29c per kilowatt/hour of power
   plus $6.55 fixed charge
   Write a program to calculate electricity charges.

9. A milkman delivers to his customers milk, cream and orange juice. The cost to customers is 65c per litre of milk, 45c per container of cream and $1.05 for 2 litres of orange juice. Write a program to calculate a person's weekly account to assist the milkman charge his customers.

10. A salesman is paid $125 per week, plus 2½% of the value of the cars he sells.
    Write a program to allow the salesman to input the values of the cars he has sold, which will then calculate his 2½% commission and finally print out the wage he will earn for that week.

Section 2.9 : REVIEW EXERCISES (4)

1. A certain tree increases its height by 1½ times per year. Write a program to calculate the height this tree will grow to in 3 years.
2. Write a program to input the time of day and to convert it to the 24 hour clock.
3. Write a program to input test scores and to print out the scores in the form of a horizontal bar graph.
4. Write separate programs to produce the following series of numbers to 10 terms

   (i) 1, 3, 9, 27 . . .

   (ii) 24, 17, 10, 3 . . .

   (iii) 1, 3, 6, 10 . . .
5. Write a program which could be used in a bank to

   (i) deposit money into an account

   (ii) withdraw money from that account

   (iii) calculate interest of 7½%
6. A man deposits $500 into a bank and leaves it there for 5 years. Each year, he is paid 11½% interest on his account and this money is added to his account.
   Write a program to calculate the amount of interest the account earns each year and also to calculate the new amount in the account for each of the 5 years.
   Note: This form of interest is called "compound interest".
7. Write a program which will calculate the compound interest on a bank account. The program should allow the user to input

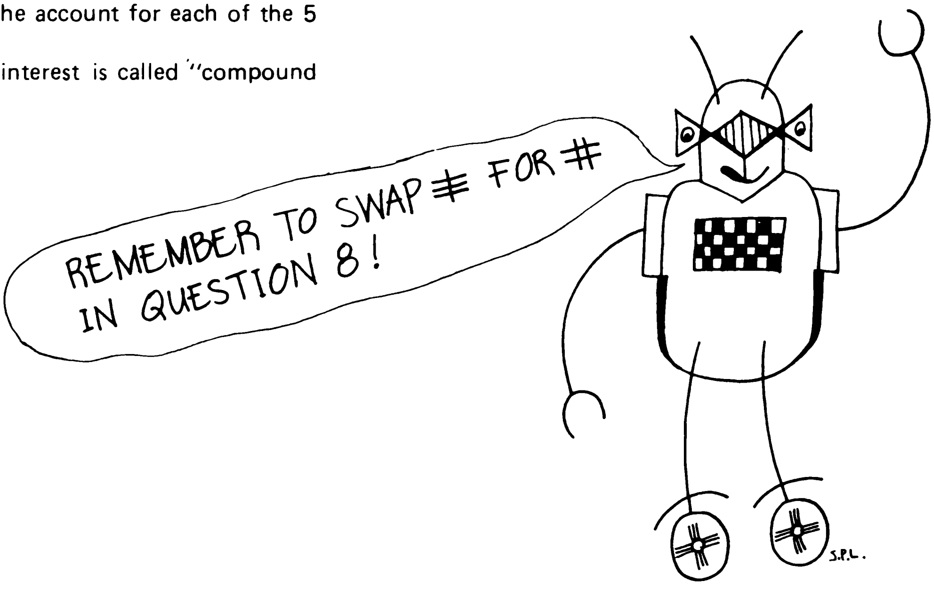
   (i) the initial amount of money (principal)

   (ii) the yearly rate of interest

   (iii) the number of years

   The output should be clearly set out in table form.
8. Write a program using nested loops and the IF — THEN statement to produce the following output.

```
        ‡
      ‡ ‡ ‡
    ‡ ‡ ‡ ‡ ‡
  ‡ ‡ ‡ ‡ ‡ ‡ ‡
‡ ‡ ‡ ‡ ‡ ‡ ‡ ‡ ‡
  ‡ ‡ ‡ ‡ ‡ ‡ ‡
    ‡ ‡ ‡ ‡ ‡
      ‡ ‡ ‡
        ‡
```


REMEMBER TO SWAP ‡ FOR # IN QUESTION 8!
S.P.L.

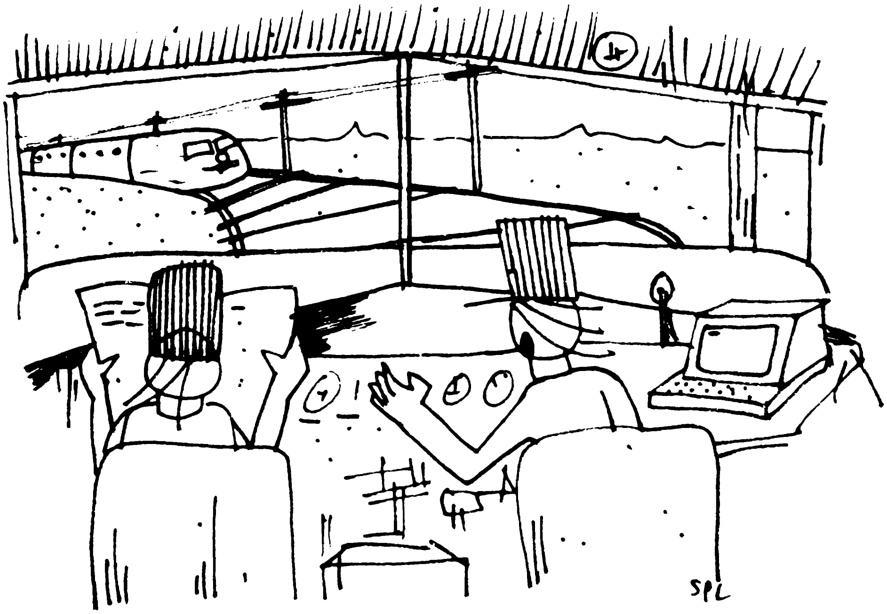
Chapter 3—PROGRAM STRUCTURE—GOOD PROGRAMMING

Programming is **not** writing down lines of BASIC code to be used to instruct the computer to do a task. This is only a small part of programming.

The major steps involved in programming may be represented in the following way:

1. Given a task, identify the major structures of the problem (i.e. what are we really being asked to find).
2. Find a method of solution suitable to the problem and computer.
3. Map out a detailed plan of the steps involved in the solution (e.g. flowcharting).
4. Convert this into code that the computer can understand.
5. Ensure that the program code can be understood by others and can be used by any likely user.

Steps 1 and 2 depend on your knowledge of the problem and experience in the use of solution techniques. We hope to give you plenty of experience throughout this book — you have already gained some. Steps 3 and 5 will be dealt with specifically in sections to follow. Of course most of the book relates to step 4.



ER, EXCUSE ME CHIEF, I THINK THERE'S AN ERROR MESSAGE HEADING THIS WAY...

Section 3.1 : READABLE PROGRAMS : THE REM STATEMENT

If we wish to put comments in our program to explain what it does but don't want them to be printed when the program is RUN we use REM statements.

All programs should contain some sort of a 'header block', the following is an example.

Example 21

```
 10  REM   PROGRAM: AVERAGE
 15  REM
 20  REM   THIS PROGRAM FINDS THE AVERAGE
 25  REM   OF A LIST OF NUMBERS
 30  REM
 35  REM   PROGRAMMER : PAUL NEWHOUSE
 40  REM   DATE : JUNE 1981
 45  REM   INITIALIZE VARIABLES
 46  LET N = 0
 48  LET S = 0
 50  PRINT "INPUT POSITIVE NUMBERS ONE PER LINE, AT END INPUT -1"
 60  INPUT A
 70  IF A = -1 THEN 110
 80  REM   ADD NUMBER TO THE SUM (S) AND INCREMENT COUNTER (N)
 90  LET S = S + A
 95  LET N = N + 1
100  GOTO 60
110  REM  FIND AVERAGE (V)
120  LET V = S/N
130  PRINT "AVERAGE IS" V
140  END
```

Lines 45, 80, 110 are comment lines to tell others what happens in the program, they only appear when you LIST the program.

Note: i) Do not put in TOO many comment lines in your programs at this stage.

ii) Always include a header block when you are going to test a program on your computer.

EXERCISE 3.1

1. When the following program is run what would be the output?

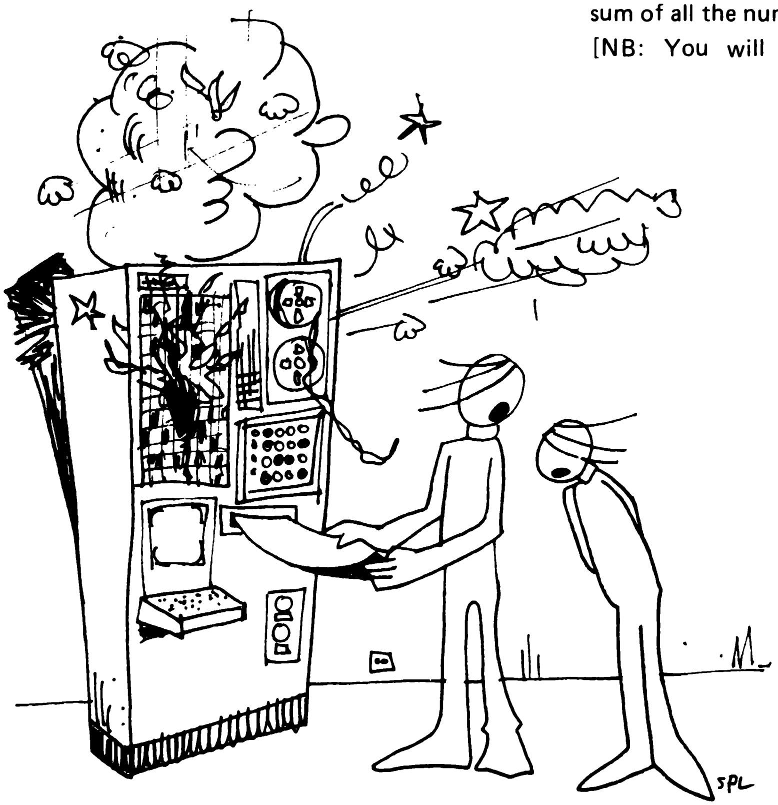
```
10 REM PROGRAM : ADDUP
20 REM THIS PROGRAM ADDS TWO NUMBERS
30 REM PROGRAMMER : FRED B
40 REM DATE : NOV 1981
50  PRINT "INPUT TWO NUMBERS WITH A COMMA BETWEEN"
60 INPUT A, B
65 REM ADD THE NUMBERS
70 LET C = A + B
80 PRINT "THE SUM IS" C
90 END
```

2. Design a program to output the product of any two numbers, include a header block.

3. Put a header block on this program (after you have worked out what it does) and rewrite it with one or two comment lines included.

```
 40 LET L = 0
 45 LET S = 100
 50 PRINT "INPUT NUMBERS SMALLER THAN 100 ONE PER LINE AT END TYPE 100"
 55 INPUT X
 60 IF X = 100 THEN 100
 70 IF X > S THEN 80
 75 LET S = X
 80 IF X < L THEN 90
 85 LET L = X
 90 GOTO 55
100 PRINT L, S
110 END
```

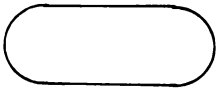
4. Design a program to act as a calculator. That is the user inputs a number, then a '+' symbol, then another number and so on until she inputs an '=' symbol at which point she is supplied with the sum of all the numbers she has input.
[NB: You will need two INPUT statements.]



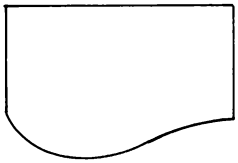
IT'S A SUICIDE NOTE!

Section 3.2 : FLOWCHARTS

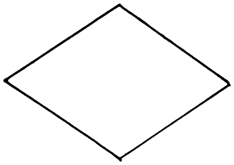
When programming far more complicated problems it is imperative that the programmer 'maps out' the steps to her solution in some way before coding it into one of the computer languages. Some ways of 'mapping out' a solution involve writing the steps down logically in english sentences or statements (sometimes called pseudo-code). However one of the most useful and easiest to follow is the method called flowcharting. The following symbols are used:



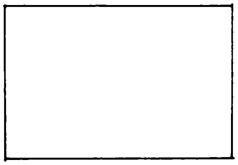
START



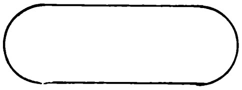
OUTPUT (Print)



MAKE A DECISION



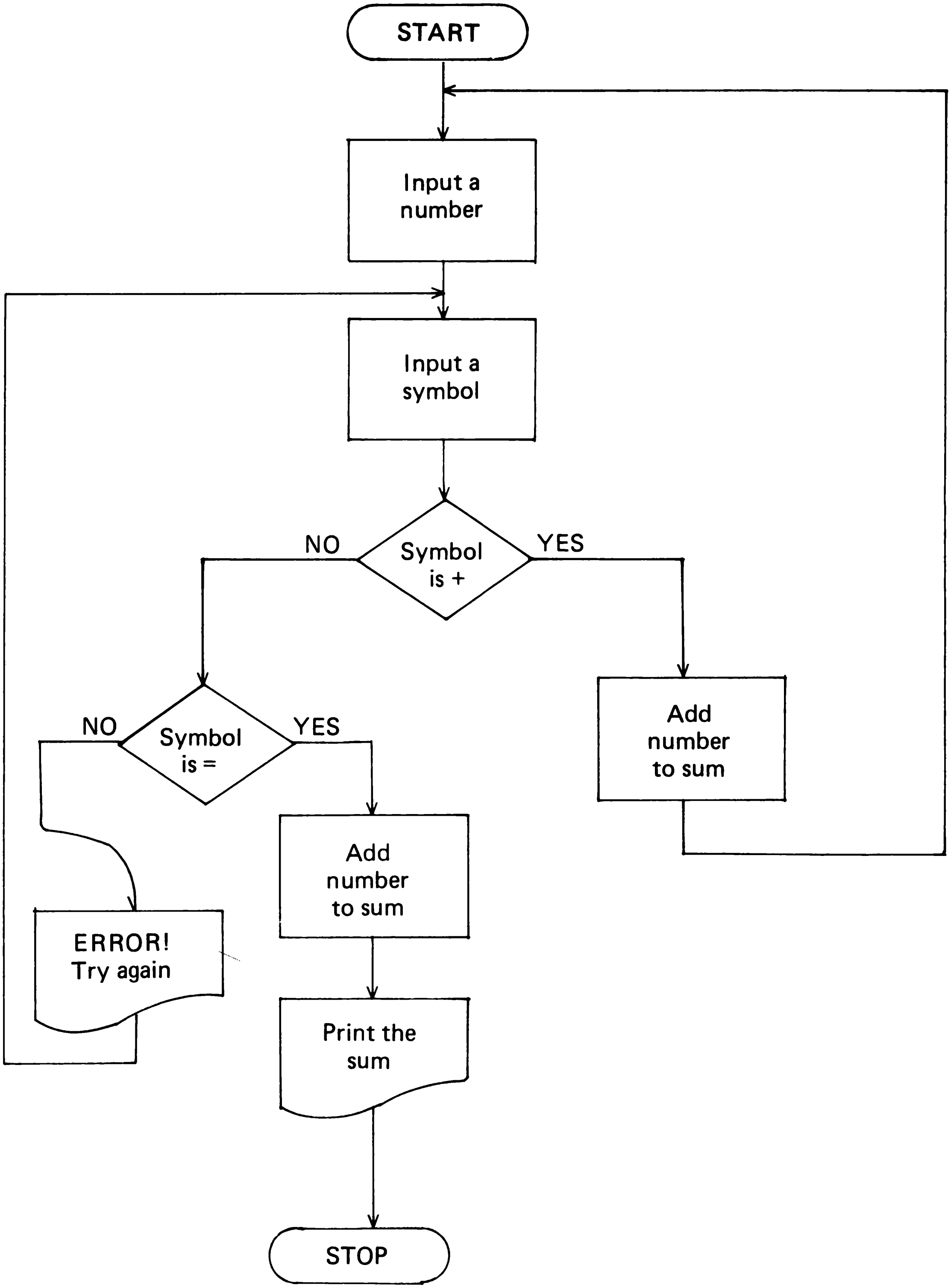
PROCESS (anything else)



STOP

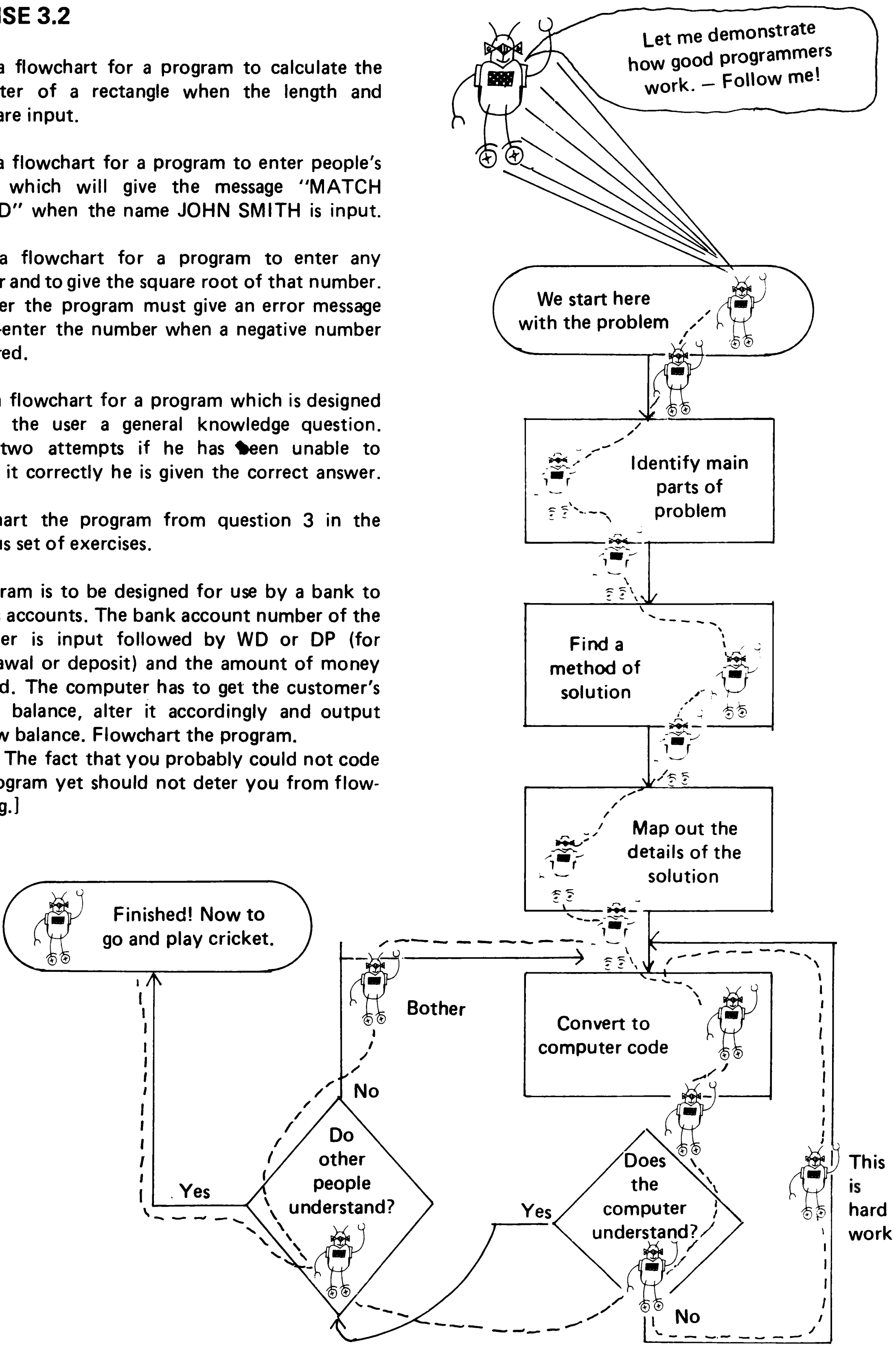
Example 22

Here is the flowchart for question 4 from the previous set of exercises (i.e. the calculator question).



EXERCISE 3.2

1. Draw a flowchart for a program to calculate the perimeter of a rectangle when the length and width are input.
2. Draw a flowchart for a program to enter people's names which will give the message "MATCH FOUND" when the name JOHN SMITH is input.
3. Draw a flowchart for a program to enter any number and to give the square root of that number. However the program must give an error message and re-enter the number when a negative number is entered.
4. Draw a flowchart for a program which is designed to ask the user a general knowledge question. After two attempts if he has been unable to answer it correctly he is given the correct answer.
5. Flowchart the program from question 3 in the previous set of exercises.
6. A program is to be designed for use by a bank to process accounts. The bank account number of the customer is input followed by WD or DP (for withdrawal or deposit) and the amount of money involved. The computer has to get the customer's present balance, alter it accordingly and output this new balance. Flowchart the program.
   [**Note:** The fact that you probably could not code this program yet should not deter you from flowcharting.]


Let me demonstrate how good programmers work. – Follow me!
We start here with the problem
Identify main parts of problem
Find a method of solution
Map out the details of the solution
Convert to computer code
Does the computer understand?
Yes
No
This is hard work
Do other people understand?
No
Bother
Yes
Finished! Now to go and play cricket.

Section 3.3 : TAB STATEMENT

The TAB statement in BASIC is used to help format computer output by allowing the programmer to output data in any position on a line.

It is always used with a PRINT statement and is of the form:

```
10 PRINT TAB (N) "......"
```

N is the number of spaces from the start of the line where the message is to be commenced. Multiple use of the TAB statement within one line is permissible.

Example 23

```
10 PRINT TAB (5) "*" ; TAB (10) "!" ; TAB (15) "?"
20 PRINT TAB (5) "COST" ; TAB (10) "TOTAL"
```

will result in, (- indicates a space)

```
----*----!----?
----COST-TOTAL
```

Example 24

```
10 FOR I = 1 TO 5
20 PRINT TAB (I) I
30 NEXT I
```

will cause

```
1
-2
--3
---4
----5
```

Interesting designs and graphs can be printed using TAB as in the previous example.

Note: On some computers, the line 20 PRINT TAB (5) "*" would yield output,

-----* (instead of ----*)

The message is printed starting in the 6th position. Check to see how your computer operates.

EXERCISE 3.3

1. Show the output from the following program segments. (Use - to show spaces)

(i)
```
10  PRINT TAB (7) "*"
```

(ii)
```
20  PRINT TAB (4) 5 * 6
```

(iii)
```
30  PRINT TAB (5) "*" ; TAB (12) "↑"
```

(iv)
```
40  LET A = 4
50  PRINT TAB (A) A ; TAB (2 * A) 2 * A
```

(v)
```
60  LET A = 4
70  LET B = 6
80   PRINT TAB (B–A) "*" ; TAB (A) "*";
    TAB (B) "*"
```

(vi)
```
 90  PRINT TAB (5) "COST" ; TAB (12)
     "TOTAL"
100   PRINT TAB (5) "++++" ; TAB (12)
     "+++++"
```

(vii)
```
110  FOR I = 5 TO 10
120  PRINT TAB (I * 2) "0"
130  NEXT I
```

(viii)
```
140  FOR J = – 4 TO 4
150  PRINT TAB (J ↑ 2) "."
160  NEXT J
```

2. Write a program segment to produce the following (remember "-" means a space)

(i)
```
----*--*
```

(ii)
```
----5-7-9
```

(iii)
```
---------VALUE-------AMOUNT
```

(iv)
```
---"
----"
-----"
------"
```

(v)
```
* * * * * * *
*---------*
*---------*
*---------*
* * * * * * *
```

3. (i) Write a program to produce the following (use TAB)

```
      *
      *
      *
   *     *
  *       *
  *       *
 <*       *>
  *       *
  *       *
  * * * * *
```

(If you are very creative, you could design your own rocket).

Using the above program, add the necessary code to:

(ii) Print out underneath the rocket

```
10  9  8  7  6  5  4  3  2  1  BLAST OFF!
```

(iii) To make the rocket "blast off", add a segment to your program to print a smoke trail below the rocket's tail pipe.

(iv) To give the impression of the rocket travelling around the world, put a GO TO statement immediately preceding the END statement which will cause the program to repeat endlessly.

4. Write a program to draw a large square.

5. Write a program to draw a large parallelogram.

6. Write a program to produce the following:

```
+--------------------------------------------+
|               ACE REMOVALS                 |
|                                            |
| NAME                           DATE        |
|                                            |
| ADDRESS                                    |
|                                            |
| AMOUNT OWED                                |
|                                            |
|                           J. MURPHY        |
|                           (MANAGER)        |
+--------------------------------------------+
```

7. Write a program to produce a table showing a conversion of the fractions 1/8, 2/8, ... 7/8, 8/8 to decimals. The program should present the table with columns and headings and a border.

Section 3.4 : GOSUB STATEMENT

It is often helpful to break a program up into smaller segments. This may be for ease in reading the program or it may be that certain segments of the program are required at more than one place in the program. These program segments are called subroutines, they may start with any statement but must end with a RETURN statement.

To use a subroutine where it is required we use a GOSUB 'n' statement in the program where 'n' stands for the line number of the first line of the subroutine. At the GOSUB statement the program jumps to the subroutine, executes the statements in it and upon encountering the RETURN statement jumps back to the next statement after the GOSUB.

Note: The subroutine must not contain a GOSUB or any statement which allows a jump out of the subroutine.

Example 25

```
 30  DIM A (20)
 40  REM ** MAIN PROGRAM **
 50  GOSUB 300
 60  FOR I = 1 TO 20
 70     LET Y = A (I)
 80     GOSUB 200
 90  NEXT I
100  STOP
200  REM ** THIS SUBROUTINE ONLY PRINTS A NUMBER IF IT IS EVEN **
210  M = INT (Y/2)
220  IF Y/2 < > M THEN 240
230  PRINT Y
240  RETURN
300  REM ** THIS SUBROUTINE FINDS THE FIRST 20 SQUARE NUMBERS **
310  FOR I = 1 TO 20
320     A (I) = I * I
330  NEXT I
340  RETURN
350  END
```

Note: 1. The main program ends with the STOP statement, the subroutines follow this, and the whole program ends with END.

2. The subroutine at line 300 is 'called' once whereas the one at line 200 is called twenty times.

3. In this program an array is used A(I) to store 20 numbers – see page 53 for a brief description of arrays.

EXERCISE 3.4

1. (i) What is output by the following program code if a 6 is input?

   (ii) What is output if a −1 is input and then a 2?

   (iii) Line 18 is perfectly legal but what problem may occur after this line (remember M is not wanted to be less than zero)?

```
  5  LET P = 1
 10  LET N = 0
 15  PRINT "INPUT A NON NEGATIVE
       NUMBER"
 16  INPUT M
 18  IF M < 0 THEN GOSUB 600
 20  GOSUB 400
 30  LET N = N + 1
 40  GOSUB 500
 50  IF N < M THEN 30
 60  STOP
400  PRINT "HELLO"
410  PRINT "THIS PROGRAM PRINTS A
       FACTORIALS TABLE"
420  RETURN
500  LET P = P * N
510  PRINT N, P
520  RETURN
600  PRINT "ERROR! TRY AGAIN – NUM-
       BER MUST NOT BE NEGATIVE."
610  INPUT M
620  RETURN
630  END
```

2. Write a header block for the type of example presented earlier in this section.

3. Write a program using GOSUB statements to be used to find the area of triangles, circles, and rectangles. The user must firstly indicate which shape he is interested in and then the program should jump to a subroutine which calculates the area for that shape.

4. Write a program to use the "IF . . . THEN GOSUB . . . " command to add, subtract, multiply or divide any two numbers the user types in. She will need prompts to ask for the two numbers and which operation she wants to use (i.e. a sub-routine for each operation).

5. The formula for the 'escape' velocity (speed) of an object leaving a planet or star is $V_e = \sqrt{\frac{2GM}{R}}$ where

   $G = 6.67 \times 10^{-11}$

   R = distance in metres

   Write a program, using this formula applied to our SUN (i.e. $M = 1 \times 10^{30}$ kg) that will do the following:

   a) Calculate V_e for values of R between 10,000km (i.e. 10^7m) and 1000000km (i.e. 10^9m), store these in an array V(100). Output these values in a table.

   b) Use these calculated values of V_e to plot a graph (using TAB) of V_e against R.

   c) Finally find R for $V_e = 3 \times 10^8$ m/sec (i.e. speed of light).

   [Use a subroutine for each part of this question.]

Section 3.5 : PROJECT (1)

Design a program to be used by teachers in writing reports. The program should:

1. Enter the student's name, class group, subject, teacher's name and the date of the report.
2. Have a selection of comments 1 to 5 representing good to bad comments for a teacher to select from.
3. Output the report in a suitable form

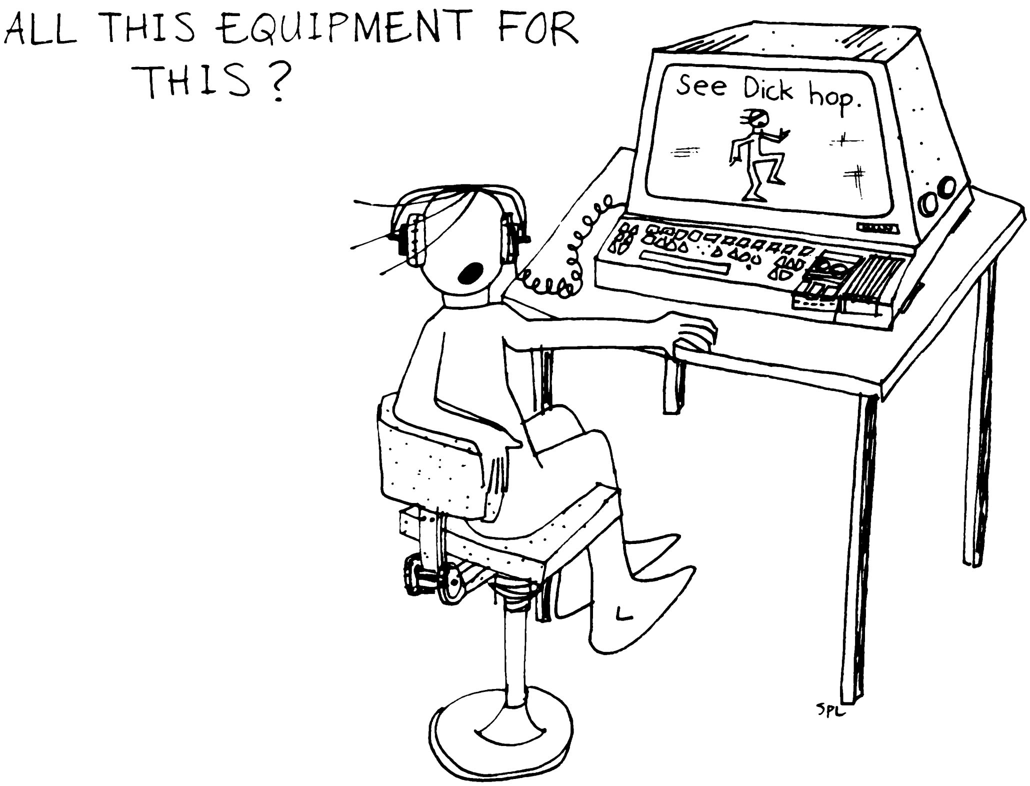
e.g. FREEMOUNT SENIOR HIGH SCHOOL

Name: B. Bloggs Class: 15
Subject: Maths

Bill has done poorly in **Maths** this term and could do better with more effort.

Teacher: P. Newhouse Date: 20/10/81

Note: **Bill** and **Maths** are inserted into the rest of the comment by the program.


ALL THIS EQUIPMENT FOR THIS?
See Dick hop.
SPL

Chapter4—SIMULATION—PROGRAMMING APPLICATIONS

Section 4.1 : THE INT FUNCTION

The INT function is used in programming to change any number with decimal places into its integer value.

The correct form is INT (number).

The INT function removes all decimal places and leaves only the greatest integer value less than the original decimal number.

e.g.

(i) PRINT INT (6.995) will output 6
(ii) PRINT INT (–3.952) will output –4

Example 26

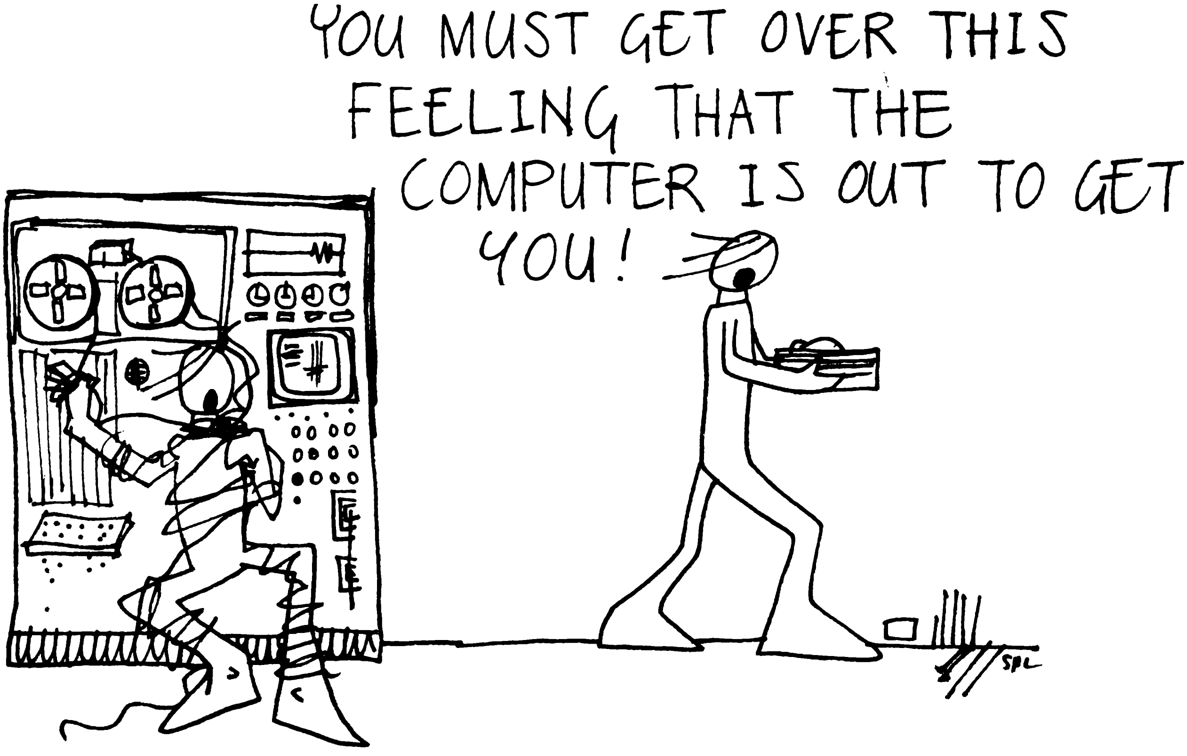
The following program uses the INT function to see if a number is even:

```
10  PRINT "ENTER A NUMBER"
20  INPUT A
30  IF INT (A/2) *2 = A THEN 60
40  PRINT "NOT EVEN"
50  GO TO 70
60  PRINT "EVEN"
70  END
```

Check that the program works by substituting different values for A.

EXERCISE 4.1

1. What are the values of the following expressions?
   - (i) INT (5.6)
   - (ii) INT (4.999)
   - (iii) INT (–2.001)
   - (iv) INT (10.9) * 2
   - (v) INT (5/2)
   - (vi) INT (7 + 8/3)
   - (vii) INT (6/3) * 3
   - (viii) INT (5/4) * INT (12/5)
   - (ix) INT (INT (7/2) * 3)
   - (x) INT (SQR (20))
2. Write a program to see if number input is divisible by 7.
3. Write a program to input any number and to see if it is a multiple of 10.
4. Write a program to input any number less than 100 which will state whether it is an odd or even number.
5. Write a program to input a number which will then divide this number by a second number and print out the quotient and remainder
   e.g. 27 ÷ 5 = 5 remainder 2.
6. Write a program to find 17% of any amount of money and will give the answer correct to the nearest cent.
7. Write a program to input a whole number less than 100 which will see if it is prime by seeing if it is divisible by 2, 3, 5 or 7. (Non primes divide evenly by these numbers).


YOU MUST GET OVER THIS FEELING THAT THE COMPUTER IS OUT TO GET YOU!
SPL

Section 4.2 : THE RND FUNCTION

The RND function is used to create random numbers between 0.000001 and 0.999999. To create a random number, one uses the form

```
PRINT RND
```

This will output a number like 0.642523.

To convert random numbers into a more useful form, the INT function is used,

e.g.

(i) PRINT INT (10 * RND) will output any integer from 0 to 9
(ii) PRINT INT (5 * RND) + 1 will output any integer from 1 to 5

Note: To ensure that a program will not create the same random numbers on separate occasions, the line

```
10 RANDOMIZE
```

is used in programs using the RND function.

EXERCISE 4.2

1. What would be the numbers generated by the following lines of code?
   In each case, assume RND = 0.524371 :
   - (i) 10 * RND
   - (ii) INT (RND)
   - (iii) INT (5 * RND)
   - (iv) INT (10 * RND) + 1
   - (v) INT (10 * RND) ↑ 2

   in (vi) – (x) give the range of possible numbers (RND has no specified value)
   - (vi) INT (RND)
   - (vii) INT (5 * RND)
   - (viii) INT (6 * RND) + 1
   - (ix) INT (20 * RND) * 10
   - (x) INT (RND * 3) – INT (RND * 5)
2. Write a program to create 10 random numbers.
3. Write a program to print a random number between 1 and 10.
4. Write a program to find two random numbers between 1 and 12.
5. Write a program to create two random numbers between 1 and 12 and to present these in a multiplication question to the user. The program should check that the user's answer is correct.
6. Write a program to print out 10 random numbers between 1 and 6.
7. Write a program to present 10 addition sums, using numbers between 1 and 20 to the user. The program should count the number of mistakes the user makes.
8. Write a program to create a random number between 1 and 30, which will give the user 5 chances to guess the number.
9. Write a program to select a number between 1 and 100. The user is to be given the chance to guess the number but is given the response "TOO HIGH" or "TOO LOW" with each choice. The program should count the number of guesses it takes.

Section 4.3 : COUNTING and SUMMING

Many tasks require the counting or summing of a large number of numbers. Using a loop in a program makes this very easy. The following program adds numbers and counts the number of negative numbers that are input.

Example 27

```
 10 REM INITIALIZE THE SUMMING AND COUNTING VARIABLES
 20 LET S = 0
 30 LET C = 0
 40 PRINT "INPUT NUMBERS ONE PER LINE"
 50 INPUT X
 55 REM IS THE NUMBER NON-NEGATIVE?
 60 IF X > = 0 THEN 90
 70 REM IF IS NEGATIVE, INCREMENT COUNTER
 80    LET C = C + 1
 90 REM ADD NUMBER TO SUM
100 LET S = S + X
110 PRINT "ANOTHER NUMBER? Y/N"
120 INPUT X $
130 IF X $ = "Y" THEN 50
140 PRINT "THERE ARE" C "NEGATIVES"
150 PRINT "THE SUM IS" S
160 END
```

EXERCISE 4.3

1. Write a program to add the numbers from 1 to 20.
2. Write a program to add the odd numbers from 1 to 99.
3. Write a program using RND to present the user with an addition sum involving 3 numbers. The user must enter the correct answer and the program must count the number of incorrect answers input.
4. Write a program which will input 10 numbers. The program must find the sum of all the odd numbers input and also the sum of the even numbers.
5. Write a program to count the number of 7's input when a user inputs 20 numbers between 0 and 10.
6. Design a program segment to be used to input term grades for a class of 30 students. The grades are A, B, C, D and F and the program should output the number of A's and the numbers of F's.
   [**Note:** Use 2 counters e.g. C1 for A's and C2 for F's.]
7. Write a program to be used by the weather bureau to input the amount of rainfall each month for a year. The program should output the total rainfall for the year and the average rainfall per month.
   [**Note:** Rainfall is measured in millimetres.]
8. Design a program used to input the prices of articles bought by a user in a shop. When a 0 is input it should stop, deduct 10% off the total amount and output this amount along with an average cost per article.
9. A user wishes to record the results of rolling a die 50 times. Write a program to allow her to input the number that appears on the die after each roll (i.e. either a 1, 2, 3, 4, 5 or 6). After the 50 rolls the program should tell her how many 1's, 2's, 3's, 4's, 5's and 6's she rolled.
   [**Note:** Use 6 counters such as C1, C2, C3, C4, C5 and C6.]

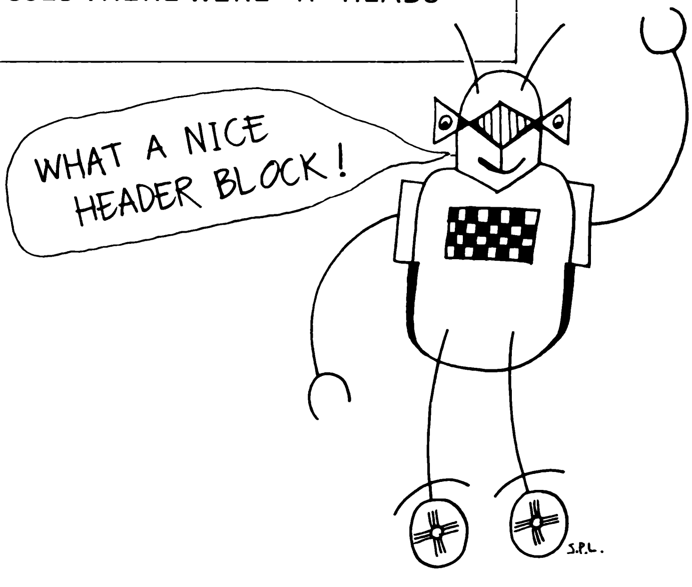
Section 4.4 : SIMULATION

On many occasions we would like to get data from experiments which are either impractical or impossible. For example experiments involving the health of people or large numbers of observations are often not practical to carry out. However we can simulate (approximate) the experiment on a computer after making a number of assumptions. This is particularly easy where probabilities are involved, as is often the case. Many 'probability' type of experiments depend on randomness and we have learnt in an earlier section how to generate random numbers.

For example let us consider the experiment of tossing a coin 100 times — this would be tedious to do, physically. Let us assume that a head or tail has the same chance of occurring each time the coin is tossed. Now if we can generate randomly 0's and 1's where they represent heads and tails respectively we can count how many of each occur. The statement X = INT (RND * 2) will supply us with only a 0 or 1 each time.

Example 28

```
 10  REM * * * A PROGRAM TO SIMULATE * * *
 20  REM * * * TOSSING A COIN ONE * * *
 30  REM * * * HUNDRED TIMES TO FIND * * *
 40  REM * * * THE NUMBER OF HEADS * * *
 50  REM * PROGRAMMER : PAUL NEWHOUSE *
 60  REM * DATE : MARCH 1981 * * * * * * * * * *
 70  REM * * * * * * * * * * * * * * * * * * * * * *
 80  RANDOMIZE
 90  LET H = 0
100  FOR I = 1 TO 100
110     X = INT (RND * 2)
120     IF X < > 0 THEN 140
130     H = H + 1
140  NEXT I
150  PRINT "IN 100 TOSSES THERE WERE "H" HEADS"
160  END
```


WHAT A NICE HEADER BLOCK!
S.P.L.

EXERCISE 4.4

1. Write a program to simulate tossing a coin. Simulate 100 tosses and print out the number of heads and tails separately. (Use two counters).

2. Write a program to simulate rolling a die. Your program should simulate 600 throws and print out the frequency each number occurs. (Use 6 counters).

3. Design a program to simulate drawing a card from a pack of playing cards (with replacement). Output the number of hearts drawn if 50 cards are drawn separately.

4. You are to design a program for the LOTTO board to pick their numbers each week to replace the use of balls.
   a) Firstly get your program to choose seven numbers in the correct range.
   b) Then modify it to check for re-occurring numbers. When a number occurs again the program should discard it and get another number. One method may be to store the numbers in a seven element array and check them using a loop.

5. What is the chance that two out of 25 randomly selected people have the same birthday? Simulate this by setting up a program which chooses 25 days randomly from the year (i.e. numbers 1 to 365). What about 23 people? Also 40 people? What is being assumed in your simulation?
   [You may only be able to print out the 25 numbers representing the days and check by eye whether any days are the same.]

6. Write a program to simulate tossing a coin 1000 times. Your program must find the largest run of heads or tails i.e. the greatest number of times heads or tails were tossed consecutively.
   (Before running this program, jot down what you would consider would be the length of the longest run and compare this with the simulated result).

Section 4.5 : REVIEW EXERCISES

1. The average temperature for each day of a month is to be input into a computer. Write a program to accept these values and output the average temperature for the month and the number of times the average for a day was above 30^0 C.

2. Write a program to output a table of twenty random integer numbers between zero and one hundred.

3. What are the values of the following expressions?
   a) INT (67.32)
   b) INT (–3.7)
   c) INT (5 * 3.2)
   d) INT (9/4)

4. Write a program to see if a number input is divisible by 11.

5. Why do we include the RANDOMIZE statement in programs needing random numbers?

6. Give the range of possible values for
   a) INT (RND + 6)
   b) INT (RND * 2) + 1

7. Write a program to test tables at random e.g. 5 x 4 = . The user should be given two tries at each question. After ten questions the program should stop and give the user his score out of ten and give him the option of another ten questions.

8. Write a program which simulates tossing two dice (use two random numbers). In fifty rolls of the pair output the number of times the sum of the two was 2, 3, 4 and so on to 12.
   [Note: You may be able to use the ON GOTO statement here.]

Section 4.6 : PROJECT (2)

A very famous simulation problem is known as the 'Barber Shop Queue'. Here we imagine a barber's shop with only one chair for haircuts and four seats for waiting customers. Let us assume that the barber takes 10 minutes to cut a person's hair and that people arrive at an average of 8 per hour (i.e. $\frac{8}{60}$ = 0.13 persons per minute). The barber's chair is free after 10 minutes from a customer sitting on it and if the waiting seats are all being used any further customers go away.

Write a program which will simulate this situation and output the length of the queue (i.e. how many waiting seats are being used) as a row of *'s, one per seat. Use a random number between 0 and 1 to indicate whether a customer comes in i.e. if RND $\leqslant$ 0.13 a customer has walked into the shop in that minute. Use a counter to keep track of the number of waiting seats occupied and if this reaches 4 no more customers are allowed to enter the shop. Also use a counter to count up lots of 10 minute to indicate when the barber's chair is free. The queue should be output every minute for an hour.

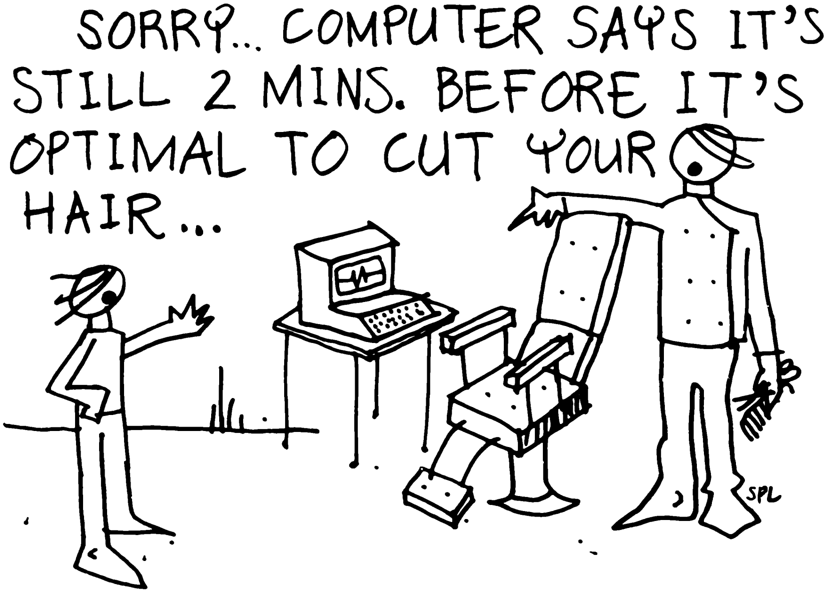
a) After getting the basic program to run see what difference is made when the number of customers per minute changes to 0.05 or 0.3.

b) What happens if the barber takes 8 or 15 minutes per haircut?

c) Add a few lines to count up how many customers are turned away each hour.

d) Alter your program so that the queue is only output every 5 minutes.

[Note: 1. You may wish to leave this program to do after the next few sections on arrays.
2. Use of subroutines will help greatly here.
3. Ask your teacher to help you to map out and understand this situation.]


SORRY... COMPUTER SAYS IT'S STILL 2 MINS. BEFORE IT'S OPTIMAL TO CUT YOUR HAIR...
SPL

Chapter 5—DATA PROCESSING—ANOTHER APPLICATION

Section 5.1 : CREATING ARRAYS

Arrays are used to efficiently store large amounts of data.

To use an array in a program, one must commence the program with a dimension statement which names the array and sets the number of memory locations to be used.

The dimension statement is of the form:

```
10 DIM A (N)      or      10 DIM A $ (N)
```

(i) N is the number of memory locations and must be a whole number.
(ii) A and A$ are sample names and allow an array to store numbers or words.

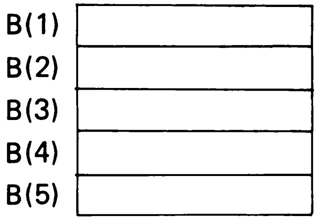
Example 29

```
10 DIM B (5)
```

establishes the array B with elements

B(1), B(2), B(3), B(4) and B(5).

Array B is represented



Arrays can be named A, B, ... Z
A1, B1, ... Z1, etc.
(similar to normal memory locations)

Example 30

Arrays are easily worked with using loops.

```
10  DIM C(6)
20  FOR I = 1 TO 6
30  C(I) = I + 2
40  NEXT I
```

This segment will establish the array and store numbers as follows . . .

C(1)	3
C(2)	4
C(3)	5
C(4)	6
C(5)	7
C(6)	8

An extra program line

```
50  PRINT C(2),  C(3),  C(5)
```

would cause the output

```
4   5    7
```

Example 31

```
10  DIM A$ (4)
20  LET A$ (1) = "YES"
30  LET A$ (2) = "NO"
40  PRINT A$ (1)
```

A$(1)	YES
A$(2)	NO
A$(3)	
A$(4)	

would establish array as indicated and output

```
YES
```

EXERCISE 5.1

1. Draw a diagram to represent the following arrays and show the contents when the following programs are run.

(i)
```
10  DIM C (5)
20  FOR I = 1 TO 5
30  LET C(I) = 2 * I
40  NEXT I
```

(ii)
```
10  DIM A$ (3)
20  LET A$ (1) = "MIKE"
30  LET A$ (2) = "JOHN"
40  LET A$ (3) = "FRED"
```

(iii)
```
10  DIM D(6)
20  FOR J = 1 TO 6
30  LET D(J) = J ↑ 2
40  NEXT J
```

(iv)
```
10  DIM A$ (3)
20  A$ (1) = "BILL SMITH"
30  A$ (2) = "21 FENTON ST"
40  A$ (3) = "MELVILLE"
```

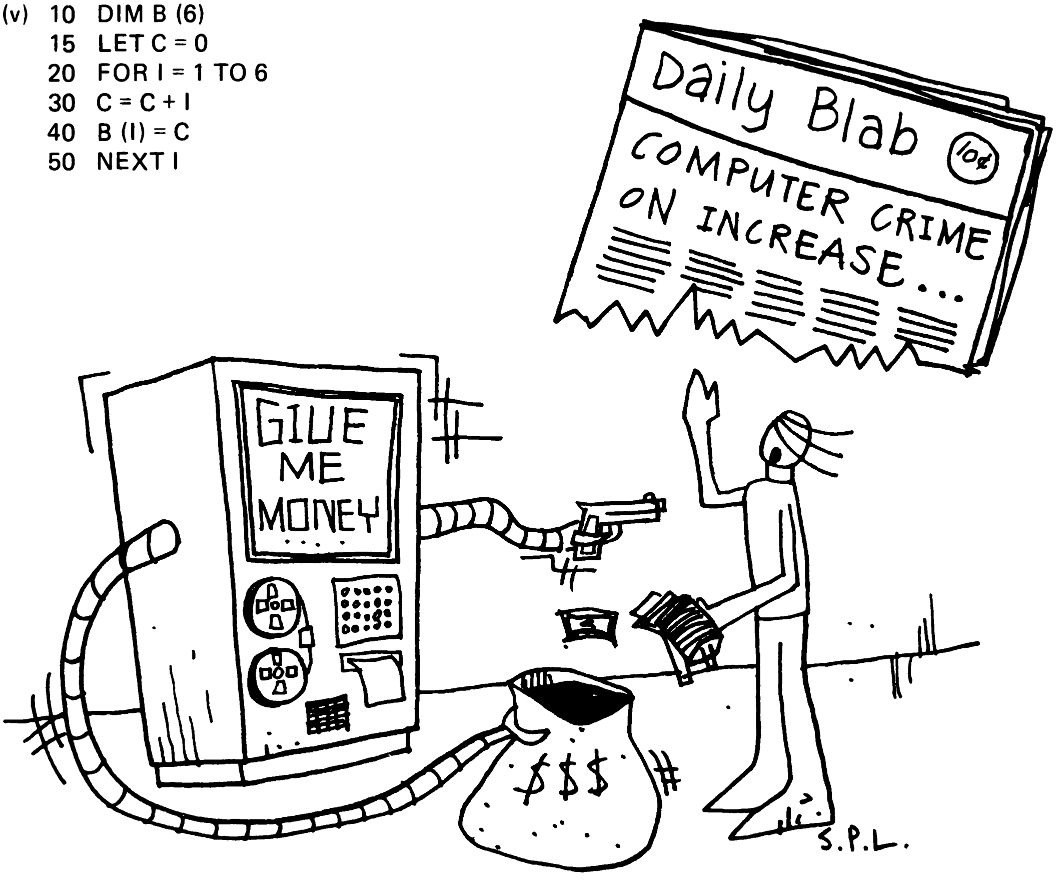
(v)
```
10  DIM B (6)
15  LET C = 0
20  FOR I = 1 TO 6
30  C = C + I
40  B (I) = C
50  NEXT I
```

(vi)
```
10  DIM A (5), A$ (5)
20  FOR J = 1 TO 5
30  A(J) = J * 6
40  A$ (J) = "J + 1"
50  NEXT J
```

(vii)
```
10  DIM A$ (5)
20  FOR S = 1 TO 5
30  LET A$ (S) = "YES"
40  NEXT S
```

(viii)
```
10  DIM A (5), A$ (5)
20  FOR B = 1 TO 5
30  LET A (B) = B
40  LET A$ (B) = "B"
50  NEXT B
```

(ix)
```
10  DIM A (10), B (10), C (10)
20  FOR I = 1 TO 10
30  LET A (I) = I
40  LET B (I) = I * 2
50  LET C (I) = I * 3
60  NEXT I
```



Section 5.2 : USING ARRAYS

As mentioned previously, arrays are suited to using loops for input and as you will see in this section, output.

Example 32

```
10  DIM X (30)
20  FOR I = 1 TO 30
30  LET X(I) = I * 2
40  NEXT I
50  FOR J = 1 TO 30
60  PRINT X(J),
70  NEXT J
```

This segment would establish the array as follows

X(1)	2
X(2)	4
X(3)	6
X(29)	58
X(30)	60

and lines 50 60 and 70 would result in,

```
2   4   6   8............56   58   60
```

being output

Example 33

```
10  DIM C2 (3), C$ (3)
20  FOR I = 1 TO 3
30  C2 (I) = 1/2 * I
40  C$ (I) = "XYZ"
50  NEXT I
60  FOR J = 1 TO 3
70  PRINT C2 (J), C$ (J)
80  NEXT J
```

would result in output

```
0.5          XYZ
1            XYZ
1.5          XYZ
```

EXERCISE 5.2

1. Show the output from the following pieces of code.

(i)

```
10  DIM A (10)
20  FOR I = 1 TO 10
30  LET A (I) = I * 10
40  NEXT I
50  FOR I = 1 TO 10
60  PRINT A (I)
70  NEXT I
```

(ii)

```
10  DIM A$ (5)
20  FOR J = 1 TO 5
30  LET A$ (J) = "HELLO"
40  NEXT J
50  FOR J = 1 TO 5
60  PRINT A$ (J)
70  NEXT J
```

(iii)

```
10  DIM C(5)
20  FOR I = 1 TO 5
30  C (I) = I + 10
40  NEXT I
50  FOR J = 1 TO 5
60  PRINT C(J), J
70  NEXT J
```

(iv)

```
10  DIM D(6), D$ (6)
20  FOR I = 1 TO 6
30  D(I) = 1/2 * I
40  D$(I) = "*"
50  NEXT I
60  FOR J = 6 TO 1 STEP -1
70  PRINT D$ (J), D (J)
80  NEXT J
```

(v)

```
10  DIM C (50)
20  FOR A = 1 TO 10
30  LET C (A) = A * 4
40  NEXT A
50  FOR B = 6 TO 10
60  PRINT C (B)
70  NEXT B
```

(vi)

```
10  DIM A (5), B (5)
20  FOR C = 1 TO 5
30  LET A (C) = C * 2
40  LET B (C) = C * 3
50  NEXT C
60  FOR J = 1 TO 5
70  PRINT A (J) * B (J),
80  NEXT J
```

2. Write a program to store the whole numbers 1 to 20 in the array A (20). Your program should also output this array.

3. Write a program to store the odd numbers from 1 to 99 in the array B (50) and to output them.

4. Write a program to input 5 numbers into an array and output them in reverse order.

5. Write a program to input 5 colours into an array. The program should output the array also.

6. Write a program to create two arrays A$ (20) and A (20). The program should allow the user to store a person's name in the array A$ and his age in the corresponding slot of array A.

7. Write a program to create two arrays S$ (5) and C$ (5). The program is to store a person's surname in S$ and his christian name in C$, in the corresponding location.
Once all names have been input, the program is to output each person's full name, christian name, followed by surname.

8. Write a program to create 3 arrays to store 5 people's names, ages and sex. The program is to output the arrays one at a time, printing out all names, ages and sex input.

9. Write a program to store the results of a test for 20 students in the array B (20). The program must calculate the average test mark.

10. Write a program to allow the user to input the results for any number of students (less than 30) into an array. The program must then allow the user to input the pass mark and the program should then print out the number of students who passed and the number that failed.

11. Write a program to store the results of a test for 30 students in an array. The program must calculate the average mark, highest mark, lowest mark and range.

Section 5.3 : READ – DATA STATEMENTS

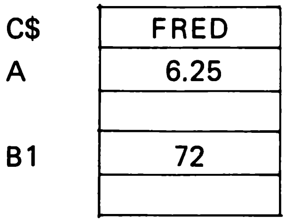
Another means of storing data in the computer is by using the READ and DATA statements. The READ statement is of the form,

```
75  READ A, B1, C$
```

and must be followed by DATA statements further in the program, in which values for A, B1, and C$ are listed. For example,

```
120  DATA  6.25
130  DATA  72, "FRED", 8
```

When a READ statement is encountered in a program, the computer searches for the next unused piece of data so in the above segment, the memory locations would be set up as,



Another READ statement occurring after line 75 would cause the next unused piece of data to be assigned. The computer keeps track of the next data item.

Example 34

```
10  READ A, B
20  LET C = A + B
30  READ A$, E$
40  PRINT A, A$, B, E$, C
50  DATA 7, 10
60  DATA "PLUS", "EQUALS"
70  END
```

would result in the following

```
7  PLUS  10  EQUALS  17
```

Example 35

```
10 DIM A$ (5)
20 FOR I = 1 TO 5
30 READ A$ (I)
40 NEXT I
50 PRINT A$ (1), A$ (3), A$ (5)
60 DATA 6, 8, 10
70 DATA 12, 16, 20
80 END
```

would result with

```
6  10  16
```

It is legal to have an excess of data in a program but insufficient data will cause an error to occur.

EXERCISE 5.3

1. Show the output of the following programs.

(i)

```
10 READ A, B, C
20 PRINT C, B, A
30 DATA 5, 6, 7
40 END
```

(ii)

```
10 READ A$
20 READ B$
30 PRINT A$
40 DATA "HELLO, "THERE"
50 END
```

(iii)

```
10 READ A, B$
20 PRINT B$
30 READ C$, B, E
40 PRINT E, B
50 PRINT C$
60 DATA 5, "+", "*"
70 PRINT "FINISH"
80 DATA 6.2, 4
90 END
```

(iv)

```
10 DIM A (3)
20 FOR I = 1 TO 3
30 READ A (I)
40 PRINT A (I)
50 NEXT I
60 DATA -1, -2
70 DATA -3, -4
80 END
```

(v)

```
 10 DIM B (5), B$ (5)
 20 FOR K = 1 TO 5
 30 READ B$ (K)
 40 READ B (K)
 50 NEXT K
 60 FOR J = 1 TO 5
 70 PRINT B (J), B$ (J)
 80 NEXT J
 90 DATA "ONE", 1, "TWO"
100 DATA 2, "THREE", 3, "FOUR", 4
110 DATA "FIVE", 5
120 END
```

2. Write a program using READ and DATA to add the numbers 107.5, 623.5 and 575.

3. Write a program using READ and DATA to multiply 672.5 and 301.6.

4. Write a program using READ and DATA to find the average of 62, 75, 37, 42 and 30.

5. Write a program using READ and DATA to find the square and square root of 90, 85, 70 and 60.

6. Write a program using READ and DATA to store the numbers 10, 20, 30, 40 and 50 in an array and to print these out in reverse order.

7. Write a program using READ and DATA statements to evaluate T and S where

   (i) $T = ar^n$

   (ii) $S = \frac{a\,(1-r)^n}{1-r}$

   $a = 12,\ r = \frac{3}{4},\ n = 5$

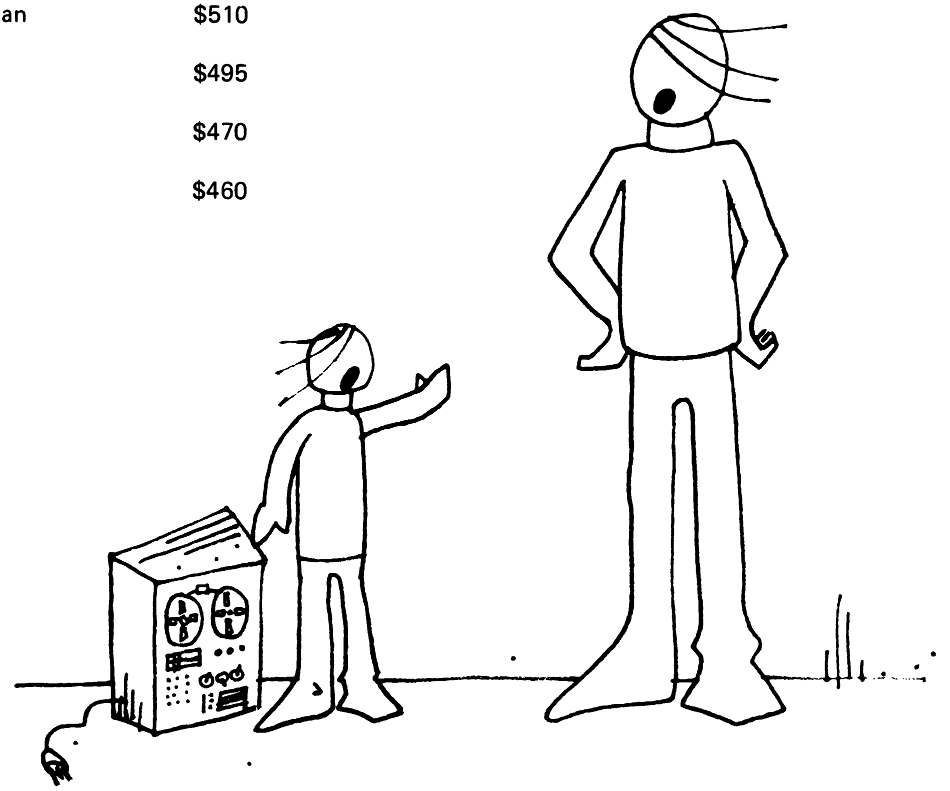
8. Write a program using READ and DATA to store 5 people's surnames in the array A$, their christian names in B$ and ages in C. The program must then output each person's full name and age and finally the average age of the 5 people.

9. A company selling swimming pools has 4 designs from which people can choose. The prices of the pools increase regularly but the profit the company makes on each pool is constant. The pools and corresponding profits are

(i) Mediterranean	$510
(ii) Pacific	$495
(iii) Indian	$470
(iv) Atlantic	$460

   Write a program using READ and DATA statements to assist the company's accounting. The program must allow the user to enter the current prices of the 4 pools, the numbers of each pool sold and the current month and year. The program must then calculate the total value of the pools sold and the profit the company will make.

10. A new car dealer sells a brand of car with 6 different models. He requires an inventory type program for his computer to assist with keeping details of stock.
    The program must store the names of each particular model together with the selling price and the number of cars of each model in stock. The program should then output the total number of cars in stock and their current market value. Write a program using READ and DATA statements to the above specifications.



IT FOLLOWED ME HOME
CAN I KEEP IT?

Section 5.4 : 2 DIMENSIONAL ARRAYS

In BASIC the statement

```
10  DIM A (5, 3)
```

creates an array with the locations as follows,

	1)	2)	3)
A(1,			
A(2,			
A(3,			
A(4,			
A(5,			

A (4, 3)

The array has 15 locations and each location is referenced by A (r, c) where r is the row and c is the column.

To read and write data with these arrays, requires normally, the use of nested loops.

Example 36

```
10  DIM B1 (4, 3)
20  FOR J = 1 TO 4
30  FOR K = 1 TO 3
40  LET B1 (J, K) = J * K
50  NEXT K
60  NEXT J
```

	1)	2)	3)
B1 (1,	1	2	3
B1 (2,	2	4	6
B1 (3,	3	6	9
B1 (4,	4	8	12

Single locations can be referenced individually also

```
70  LET B1 (2, 3) = 6          etc.
```

Example 37

Given the array C$ (3, 2)

ONE	TWO
THREE	FOUR
FIVE	SIX

```
100  FOR I = 1 TO 3
110  FOR J = 1 TO 2
120  PRINT C$ (I, J),
130  NEXT J
140  NEXT I
```

would produce output

```
ONE  TWO  THREE  FOUR  FIVE  SIX
```

To output the array in the normal represented form one would need to include the line

```
135  PRINT
```

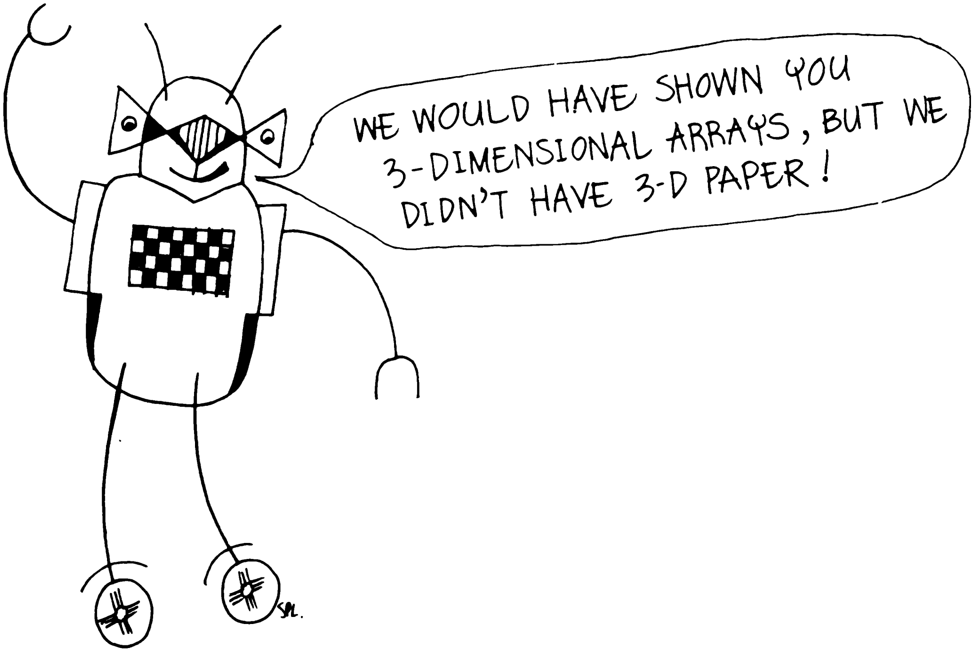
* NOTE:

On some versions of BASIC, there exist MAT commands to assist when dealing with 2 dimensional arrays.

For example

```
50  MAT PRINT A
60  MAT READ A
```

allows the user to input and output complete arrays without the use of nested loops. Check to see if your computer uses the MAT commands because they can be very useful.


WE WOULD HAVE SHOWN YOU 3-DIMENSIONAL ARRAYS, BUT WE DIDN'T HAVE 3-D PAPER!

EXERCISE 5.4

1.

1	13	14	15	8
12	2	16	9	7
11	18	3	19	6
17	10	20	4	5

This table represents the contents of array B.

(i) Complete the following statement for a program containing array B.

```
10 DIM B ( , )
```

(ii) List the numbers stored in the following locations,

a) B (2, 5)

b) B (3, 1)

c) B (4, 4)

d) B (1, 2)

(iii) Write down the location containing the following numbers,

a) 12

b) 6

c) 15

d) 5

(iv) Write a program segment to output array B in table form.

2.

HOLDEN	SEDAN	1973
FORD	UTE	1976
TOYOTA	LIFTBACK	1980
DATSUN	COUPE	1979
SIGMA	WAGON	1977

The above table represents the contents of array B$.

(i) Write the first line of a program containing this array.

(ii) List the output from the following,

```
40  PRINT B$ (1, 2)
50  PRINT B$ (2, 3)
60  PRINT B$ (4, 3)
```

(iii) Write the output from the following,

```
70  FOR J = 1 TO 5
80  PRINT B$ (J, 3)
90  NEXT J
```

(iv) Rewrite the array B$ and include the following changes

```
100  LET B$ (4, 1) = "SUBARU"
110  LET B$ (4, 2) = "SEDAN"
120  LET B$ (4, 3) = "1975"
```

3. Examine the following array representations.

A

20	16	12	8	4
19	15	11	7	3
18	14	10	6	2
17	13	9	5	1

A$

A	B
C	D
E	F
G	H
I	J
K	L

(i) Write a dimension statement for each array.

(ii) Write down the locations of,

a) 15
b) 6
c) G
d) H

(iii) What is stored in the locations,

a) A (2, 3)
b) A$ (1, 2)
c) A (4, 1)
d) A$ (2, 2)

(iv) What is output by the following segments?

a)
```
100  FOR I = 1 TO 5
110  PRINT A (2, I)
120  NEXT I
```

b)
```
130  FOR J = 1 TO 6
140  PRINT A$ (J, 1)
150  NEXT J
```

c)
```
160  FOR K = 1 TO 2
170  PRINT A (K, K), A$ (1, K)
180  NEXT K
```

(v) Write a program segment to output,

a) 18, 14, 10, 6, 2

b) B D F H J L

c)
```
4
3
2
1
```

4. Draw a diagram to represent the contents of the following arrays.

(i)
```
10  DIM A$ (3, 2)
20  FOR J = 1 TO 3
30  FOR K = 1 TO 2
40  READ A$ (J, K)
50  NEXT K
60  NEXT J
70  DATA "JAMES", "JOHN", "HARRY"
80  DATA "BILL", "FRED", "MIKE"
```

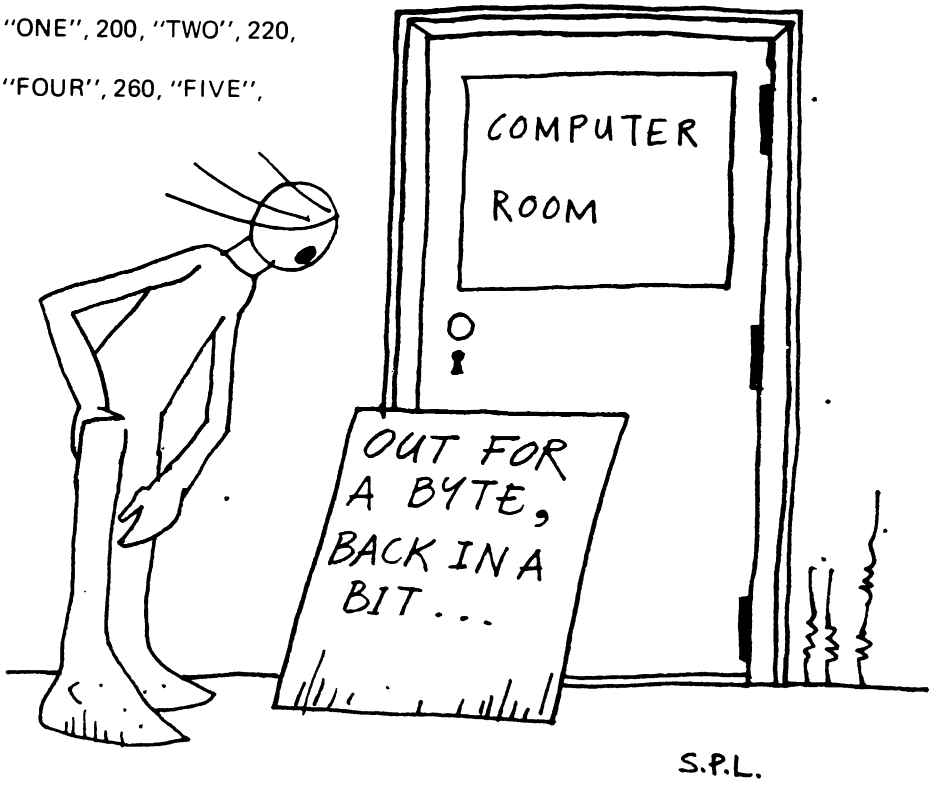
(ii)
```
10  DIM A (5, 2),  C$ (5, 2)
20  FOR I = 1 TO 3
30  FOR J = 1 TO 2
40  READ A (I, J),  C$ (I, J)
50  NEXT J
60  NEXT I
70  DATA 180, "ONE", 200, "TWO", 220,
    "THREE"
80  DATA 240, "FOUR", 260, "FIVE",
    280, "SIX"
```

5. Write a program to create and print out the array D (5, 6) which will contain the number 5 in every location.

6. Write a program to create and print out the array E$ (4, 2) which will contain the word "HELLO" in every location.

7. Write a program to create an array which will store 5 people's names (christian and surnames are to be stored independently). Your program should output the surnames and christian names separately.

8. Write a program to store 6 students names and the marks from 4 tests. (Use an INPUT statement and two arrays). The program should output the results in table form with the heading.


COMPUTER ROOM
OUT FOR A BYTE, BACK IN A BIT...
S.P.L.

Section 5.5 : MANIPULATING ARRAYS (1) : FINDING THE LARGEST OR SMALLEST ELEMENTS:

Given an array X(N) (or X$(N)) the following lines of code will select the largest and smallest values and place these in the variables L and S.

Example 38

```
100  LET S = X (1)            }initialise with the
110  LET L = X (1)            }first element
120  FOR I = 2 TO N           }
130     IF X (I) < L THEN 150
140        LET L = X (I)      } new largest value
150     IF X (I) > S THEN 170
160        LET S = X (I)      } new smallest value
170  NEXT I
```

EXERCISE 5.5

1. If the following values are stored in an array B(N) in the order 5, 6, 10, –2, 15, 4 (i.e. N = 5) what will be the value of P at line 100 and line 150 in this piece of code?

```
100  LET P = B (1)
110  FOR I = 2 TO 5
120     IF B (I) < P THEN 140
130        LET P = B (I)
140  NEXT I
150  PRINT P
```

2. Write a program which allows the user to input 10 numbers. The user should in return be supplied with the value of the smallest number by your program.

3. Write a program which stores names in an array A$(N) and outputs the, alphabetically, first and last names.
   [Note: Use the same algorithm as for Example 38 for A(N) as the computer orders alphabetically in much the same way as it orders numerically.]

4. Using the line INPUT C$ (I), D (I) write a program which stores the names of 25 employees of a company and their wages. The program should output the highest wage and the name of the highest wage earner.

Section 5.6 : MANIPULATING ARRAYS (2) : SORTING

Earlier you wrote a program to sort 3 numbers from largest to smallest. This was a fairly complex program so I think you can see that this method is unsuitable for any larger number of numbers.

There are many sorting algorithms that can be used, most of them based on loops, here is one called the BUBBLE – SORT.

Example 39

To order numbers stored in array A(N).

```
200  FOR I = 1 TO N – 1
210    FOR J = I + 1 TO N
220      IF A(I) > A(J) THEN 260
230        T = A(I)           } swap A(I)
240        A(I) = A(J)        } and A(J) using
250        A(J) = T           } a temporary variable
260    NEXT J
270  NEXT I
```

When you complete question 1 you should notice that this algorithm really finds the largest value, stores it in A(1) then finds the next largest, stores it in A(2) and so on.

EXERCISE 5.6

1. For the array A(4)
   - (i) Rewrite the lines of code of the algorithm for N = 4.
   - (ii) Copy the array boxes down as they are below.
   - (iii) Write down in the array boxes what is stored in the array positions after each pass through the inside loop (i.e. at line 260). Keep track of the value of the counters I and J to help you.

	A						
A(1)	6						
A(2)	10						
A(3)	5						
A(4)	8						
I							
J							

2. Using the coding from the BUBBLE–SORT algorithm in a program which stores 20 numbers in an array, sort them and output the sorted array.

3. Write a program to store 25 names and sort them into alphabetical order.
[Note: Use the same algorithm.]

4. Modify your program from question 4 in the previous section so that it will list alphabetically the employees names and their respective wages.

Section 5.7 : PROJECT (3)

USED CAR YARD INVENTORY PACKAGE

You are asked to write a program to be used by used car dealers. The program should create an array A$(50,4) to store the details of up to 50 cars in the following manner:

	1)	2)	3)	4)
A$(1,	MAKE OF CAR	BODY TYPE	YEAR OF CAR	PRICE

The program should then,

(i) read the details of 35 cars into the array.

(ii) give the user the following options:

a) list all cars in stock
b) allow the user to input any make of car and then to list all cars of that make
c) allow the user to input any body type and then to list all cars of that body type
d) allow the user to input any year and then to list all cars of that year
e) to insert the details of a new car
f) to delete the details of a car which is sold

Output should be clear and concise and the program should be structured carefully.

Note: To assist all students writing the program, it would be very helpful to compile the data of the 35 cars into a program called CARS, which would be comprised solely of DATA statements commencing at line 1000.

When students are ready to enter their programs, they could use the OLD command to bring the program CARS into main memory and then to simply add their program lines to CARS.

Each student could then use the SAVE command renaming the ammended program CARS as their own.

Section 5.8 : PROJECT (4)

TEST SCORE ANALYSIS PACKAGE

Design and test a program to be used by teachers in analysing test results from a class of students. Build up the package using subroutines to do each task, controlled by a main program largely consisting of a series of GOSUB statements.

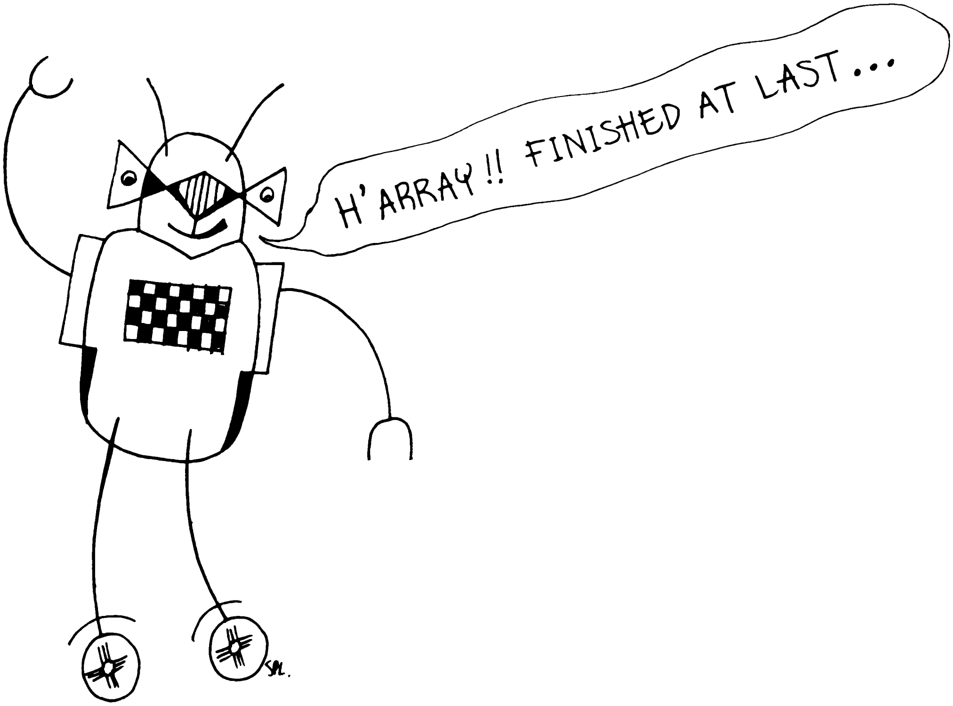
The program should

(i) allow the user to input test results,
(ii) sort the results,
(iii) calculate the mean and median,
(iv) give the highest and lowest scores,
(v) calculate the range,
(vi) calculate the mode,
(vii) output the results graphically,
(viii) calculate the standard deviation of the results (you will need to ask your teacher for help here).

Provide clear output in a manner you feel may be useful to a user.

Include anything else, which you feel may be useful. To test your package, obtain some real test scores from your teacher.

REMEMBER: Make the package easy to use.


H'ARRAY!! FINISHED AT LAST...

Chapter 6—FILES—DATA STORAGE

Section 6.1 : SEQUENTIAL FILES

A file is just a collection of items, so a program is a file. We are interested in this section in data files i.e. a file of data items. A sequential data file consists of data items which may only be read from the first to the last. They are like data stored on a tape where you can only start at the beginning of the tape and play it through until you come to the place you want. We will represent sequential files as being stored on tape BUT you must remember that sequential files may be stored on disk or any other form of memory. It is the way they are set up not the medium they are stored on that makes them sequential.

Often we combine these items into groups because they are related in some way to one another. We call each group a RECORD. For example we may want to keep student records in a file so that each record contains the student's name, his class number, his phone number and I.Q. level. Each of these items of data is known as a FIELD.

The file may be represented in the following way:

First student record				Second student record			
BILL SMITH	9	4392111	131	PAUL KING	15	6213011	125

and so on

To locate a specific student record it would be necessary to start at the beginning of the file and to examine each record until the desired record was found.

EXERCISE 6.1

1. Design a patient record to be used by hospitals, include at least four fields of your own choice to each record. Draw two records in the same way as has been done in our example above.
2. A hardware store maintains records on all stock. Identify some possible fields which would be held on different articles sold by that shop e.g. code number for article, article name.
3. Design a record structure for a library which wishes to keep a record for each book it has on its shelves.

Section 6.2 : CREATING AND USING SEQUENTIAL FILES ON A COMPUTER

One problem we encounter at this point is that each computer designed tends to have slightly different BASIC commands or statements in relation to files. This is particularly true of the microcomputers available today. So we intend to give you the general form of most file handling systems, then give you an example of a microcomputer's system (i.e. the APPLE II) and finally give you a standard BASIC statement set.

The general format of file handling can be represented in the following steps.

1. Open or identify the files to be used for input or output.
2. Indicate whether we wish to read from a file or write to it, each time we use it in the program segment.
3. Close or identify the end of the use of the files.

These steps are easily recognised in this APPLESOFT program segment.

Example 40

```
 55  LET D$ = CHR$ (4)
 60  PRINT D$; "OPEN SAMPLE 1"
 65  PRINT D$; "OPEN SAMPLE 2"
 70  PRINT D$; "READ SAMPLE 1"
 80    INPUT A$
 90  PRINT D$; "WRITE SAMPLE 2"
100    PRINT A$
110  IF A$ < > "ZZZ" THEN 70
120  PRINT D$; "CLOSE SAMPLE 1"
130  PRINT D$; "CLOSE SAMPLE 2"
```

In this code segment:

a) line 55 puts D$ equal to the control-D character (this is needed for the file handling commands in APPLESOFT).
b) lines 60 and 65 open the files.
c) line 70 indicates that we want to read a data item so the INPUT statement in line 80 is assumed to refer to getting data from the file (no ? comes up on the screen).
d) line 90 indicates that we want to write a data item so the PRINT statement in line 100 is assumed to refer to writing data to the file.
e) line 110 checks for a marker we have put at the end of the file i.e. "ZZZ"
f) lines 120 and 130 close the two files.

Here are the commands in one of the major BASIC command sets.

1. To open a file or files:

   e.g.
   ```
   60  FILES SAMPLE, OUTPUT
   ```

   – this sets up file channels where channel ‡ 1 refers to file SAMPLE and channel ‡ 2 refers to file OUTPUT.

2. To read a file or write to a file

   e.g.
   ```
   70 INPUT # 1, N, T$, Y$
   ```

   — this reads the file in channel # 1 and reads the next three items into N, T$ and Y$ respectively.

   e.g.
   ```
   90 PRINT # 2; N; T$; Y$
   ```

   — this writes the three items into the next positions in the file in channel # 2.

3. To find the end of a file and close off:

   e.g.
   ```
   100 IF END # 1, THEN 380
   ```

   — this looks to see if the end of the file in channel # 1 has been reached, if so it jumps on to line 380.

4. Two additional and useful commands are:
   a) QUOTE # 2

      — this ensures that all literal strings written to the file in # 2 are enclosed in quote (" ") marks in the file.

   b) SCRATCH # 2

      — this deletes the present file in # 2 and rewinds it to the beginning to start writing to it again.

Example 41

This is the same program as Example 40.

```
 60  FILES SMPL01, SMPL02
 70  INPUT # 1, A$
 80  PRINT # 2; A$
 90  IF END # 1, THEN 110
100  GOTO 70
110  END
```

Note:

1. To create a sequential file to start with, your system may allow you to type it in like a program.

   e.g.
   ```
   NEW STUDRD
   10 "BILL SMITH" 9 4392111 131
   20 "PAUL KING" 15 6213011 125
   and so on
   ```

2. Remember before you start to code any programs using files please refer to your particular computer's manual to check on its particular set of statements.

EXERCISE 6.2

1. Write a program to store any number of names input by a user in a sequential file.

2. Write a program to read the sequential file you have created in the previous question.

3. A company keeps employee records containing the employees name and pay scale (a number of dollars an hour). Write a program to read these records and print them out. You will have to set the file up first.
   [Hint: to read the file use INPUT ǂ 1, N$, P inside a loop to read each record – N$ for the name and P for the rate of pay.]

4. Write a program to read the list of names created in question 1 into an array, sort them using a bubble-sort and then write them back into another file. How could you make it write them back into the same file?
   [Note: You will need a counter, N, to know how many names are in your array after reading the file.]

5. Add subroutines to your program in question 3 to do each of the following tasks:
   a) Read through the file and only print out a particular employees pay rate.
   b) Allow a user to input the number of hours each employee has worked that week. The subroutine should output how much that employee should be paid for the week.

Section 6.3 : PROJECT (5)

Write a program package to be used in schools to store information on students. The data file is to contain records, one per student, consisting of the following fields : name, class group, phone number and I.Q. level.

1. Draw a diagram of the record structure.

2. Create a data file of between ten and twenty records.

3. Read the data into four arrays N$(I), T(I), P(I) and Q(I) where this represents record I in the file. Then sort the records according to name [i.e. N$(I)] – remember when you swap two names you must swap the other three array positions. Finally write the arrays back into a new file as records – they should now be in alphabetical order.

4. Include subroutines to allow students to be deleted from the file or a new student to be inserted. This is called an UPDATE of the file and will necessitate reading the file and writing the updated data to a new file.

5. Write a subroutine to be used to output details on an individual student when requested to do so.

[Note: Build your program up using subroutines and testing it after each stage.]

Chapter 7—APPENDIX

Section 7.1 : INBUILT FUNCTIONS

The BASIC language and many computers have often used functions built into the language, we have already included the RND, TAB, INT and SQR functions. Here are a few more useful functions.

a) ABS (X) finds the absolute value of the value of the expression X.
e.g. ABS(–5) = 5
and ABS (Y) = 7 if Y = 7

b) SGN(X) is +1 if X is positive and –1 if X is negative.

c) P = LOG(X) ; i.e. $P = \log_e X$
and P = EXP(X) ; i.e. $P = e^x$

d) Trigonometric functions:

SIN(X) : i.e. sine of X
COS(X) : i.e. cosine of X
TAN(X) : i.e. tangent of X
ATN(X) : i.e. arctan of X

However you must use radians with these trigonometric functions.

EXERCISE 7.1

1. Write a program to output the absolute value of any number input.

2. In quadratic equations of the form $y = ax^2 + bx + c$ the roots of the equation can be found using the formula $x = \frac{-b \pm \sqrt{b^2 - 4ac.}}{2a}$
Write a program to find the roots of any equation of this form the user wants to use. The determinant (i.e. $b^2 - 4ac$) must not be negative to give real roots. Can you include an error routine to stop the program if the determinant is negative?

3. Write a program to find the sine of any radian measurement.

4. Write a program to find the cosine of any angle in degrees. [Use the conversion Y = (X/180) * 3.1416 where X is a number of degrees.]

5. Design a program to construct trigometric tables for angles varying between -180^0 and 180^0, using increments of 10^0. The table headings should be: X[DEGREES] X[RADIANS] COS(X) SIN(X) TAN(X).

6. Add a subroutine to your program in question 5 to draw a side-on graph of SIN(X). Let a print line represent increments of 10^0 and use the TAB function to position an * along the line at an appropriate position. The output may look like this

```
  –180^0                 *
  –170^0               *
  –160^0             *



      .
   180^0                 *
  SCALE –1  –0.5  0  0.5  1
```

Include any additional improvements, such as axes, that you can think of.

7. Design further subroutines to graph the COS(X) and TAN(X) functions.

8. Write a program to evaluate y, if $y = \log_e(x) + e^x$ for any value of x.

9. A wave in physics is often defined by y = A sinBx where A and B are given constants. Write a program to graph a wave when the user inputs values for A and B.

10. Write a program to graph y = sin x [using *'s] and $y = \sin(x + 30^0)$ [using ǂ's] on the same set of axes.

Section 7.2 : THE DEF STATEMENT

When a programmer wishes to define a function, he uses the DEF STATEMENT. Consider the line of code,

```
50 DEF FNA(X) = X ↑ 2 + X ↑ 3
```

Once a function like this has been defined, it can be referred to the program as follows

```
60 PRINT FNA(2)
```

As would be anticipated, 12 would be printed.

Note: (i) Up to 26 functions can be defined in any one program by using the names FNA, FNB . . . ., FNZ

(ii) More than one argument can be used in defining a function
e.g. FNA (X, Y) = X ↑ 2 + Y ↑ 2

Example 42

```
10 DEF FNA(X) = X * 2 + 3
20 DEF FNB(Y) = Y ↑ 3
30 DEF FNC(X, Y) = X ↑ 2 + Y ↑ 2
40 DEF FND (X, Y, Z) = X * Y * Z
50 PRINT FNA(5), FNB(3), FNC(2, 3), FND(2, 3, 4)
60 END
```

would result in output,

```
     13      27      13      24

i.e. 5 * 2 + 3   3 ↑ 3   2 ↑ 2 + 3 ↑ 2   2 * 3 * 4
```

EXERCISE 7.2

1. Given the following functions

```
10  DEF  FNA(A) = A + 3
20  DEF  FNB(B) = B * 3 + 5
30  DEF FNC(C, D = (C+D)/2
40  DEF  FND(X, Y, Z) = X – Y * Z
```

evaluate

(i) FNA(5)
(ii) FNB(–2)
(iii) FNC(5, 7)
(iv) FND(2, 6, 5)

2. Write the following as user defined functions

(i) $f(x) = x^2 + 3$
(ii) $g(y) = y^3 + 2y^2 + 3y + 5$
(iii) $h(x) = \sin^2 x + \cos^2 x$
(iv) $f(x, y) = x^2 + y^2$
(v) $g(x, y) = \frac{x}{y} + 2xy$

3. Use the DEF statement in a program to convert degrees celsius to degrees fahrenheit

$$F = \frac{9}{5}C + 32$$

4. Use the DEF statement to convert percentage to a fraction.

5. Use the DEF statement in a program which calculates a percentage of an amount of money (you will need 2 arguments, percentage and amount of money).

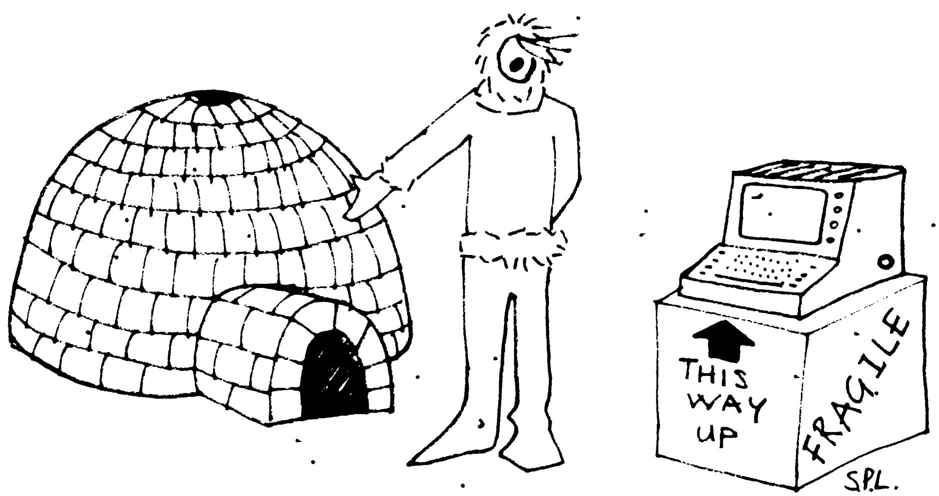
6. Use the DEF function to calculate values of sin(x) ↑ 2 + cos(x) ↑ 2 with x varying from –10 to 10.

7. Compound interest which accrues on bank accounts can be calculated using the formula

Total money in bank
= principal (1 + interest rate)$^{\text{number of years.}}$

Use DEF in a program to determine how much money will accrue using principal, interest rate and time as arguments of the function.

BOY, THAT SALESMAN SURE HAD A WAY WITH WORDS!



Section 7.3 : ON – GO TO STATEMENT

When using the IF – THEN statement only 2 branches are possible. In a program where it is necessary to branch to more than 2 segments, the ON – GO TO statement is used.

The form of this statement is as follows:

```
40  ON  A  GO TO  X, Y, Z . . . .
```

where A is a location with integer value and X, Y and Z etc. are line numbers.

If A = 1, the program branches to line X
A = 2, the program branches to line Y and so on.

There is no limit to the number of branches possible but it would not be good programming technique to use too many.

Example 43

```
 10 REM THIS PROGRAM FINDS PERCENTAGES
 20 PRINT "WHAT IS THE AMOUNT OF MONEY"
 30 INPUT A
 40 PRINT "WHAT PERCENTAGE?"
 50 PRINT "TYPE 1 – 10% "
 60 PRINT "TYPE 2 – 20%"
 70 PRINT "TYPE 3 – 30%"
 80 INPUT A
 90 ON A GO TO 100, 120, 140
100 PRINT B * 0.1
110 GO TO 150
120 PRINT B * 0.2
130 GO TO 150
140 PRINT B * 0.3
150 END
```

EXERCISE 7.3

1. A program includes the following setment

```
100  INPUT A
120  ON A GO TO 100, 160, 120, 140
```

   (i) What possible values of A would be expected?

   (ii) Which line would the program branch to if 3 was entered?

   (iii) What integer must be entered to cause a branch to line number 160?

   (iv) Rewrite line 120, so that when 5 is input, the program branches to line 200.

2. A program includes the following lines

```
20  INPUT B
30  LET C = B * 2 – 1
40   ON C GO TO 100, 150, 200, 230, 250, 310.
```

   (i) What value of C would result in a branch to line 200?

   (ii) What is the largest value of C which would result in an expected branch?

   (iii) If the number 2.5 was input for B, what number is stored in C?

   (iv) Where would the program branch to when 2.5 was input?

3. Write a program using ON – GOTO which inputs the radius of a circle and gives the user the option of finding the area or circumference ($A = \pi r^2$ and $C = 2 \pi r$)

4. Write a program using ON – GOTO which inputs the length of a side of a cube and gives the user the option of finding the surface area or volume. ($V = l^3$ and $S.A. = 6 l^2$)

5. Write a program using ON – GO TO which can find the volume of a cube, rectangular prism, cylinder or sphere. Include a segment which will ask the user "do you want another go?", in order to allow a re-run of the program.

6. Write a program to allow a teacher to input marks into an array and then gives the following option using the ON – GO TO

   a) the average mark
   b) the highest mark
   c) the lowest mark
   d) the range of the marks

7. Write a program which asks a general knowledge question and gives 4 choices for the answer. Use ON – GO TO to respond to all answers with an appropriate comment.

8. When using ON – GO TO, the number which causes the branch must be an integer and in a certain program, which has been written, an integer from 1–20 which is to be input into B. Write a program segment to check that the character input is an integer in the necessary range which will give an error message when an illegal character is input and re-execute the input.

9. Use an ON – GO TO statement in a program which inputs numbers. Any number greater than zero is stored in an array. If the user types zero, the computer returns the number of numbers that he has stored so far. If he types a –1, he gets the largest number stored. If he types a –2, he gets the smallest number stored. If he types a –3, the computer prints back all the numbers in the order that they were typed in.

SOLUTIONS

NOTES ON THE SOLUTIONS

The solutions for selected questions in each set of exercises are given below. In each case at least the first few questions have been answered. It should be noted that solutions in computer programming are in no way unique and so students may use different techniques to get the same results. The only true method to check a solution is to run it on a computer — this we suggest students do as often as possible.

EXERCISE 1.1

```
1. (i)   THE FIRST SECTION
   (ii)  THE  FIRST  SECTION
   (iii) THE
         FIRST
         SECTION
   (iv)  1    2    THREE
   (v)   1    2    THREE
   (vi)  100          200
         300

2. (i)   10  PRINT "ONE HUNDRED"
   (ii)  10  PRINT "ONE HUNDRED"
   (iii) 10  PRINT "ONE", "HUNDRED"
   (iv)  10  PRINT "ONE", 100
   (v)   30  END

3. 10  PRINT "COMPUTING IS EASY"
   20  END

5. 10  PRINT "10 9 8 7 6 5 4 3 2 1"
   20 PRINT "BLAST OFF!"
   30  END
```

EXERCISE 1.2

1. (i) the computer erases the current program in main memory and accepts a new program AREA 1.
   (ii) displays the current program in main memory.
   (iii) executes the current program.
   (iv) transfers a copy of the program in main memory to auxiliary memory.
   (v) erases the program AREA 1 stored in auxiliary memory.
   (vi) transfers a copy of AREA 2 from auxiliary memory to main memory.

```
2. (i)   LIST
   (ii)  RUN

   (iii) SAVE
   (iv)  UNSAVE  VOL 1
   (v)   OLD  VCL 2
   (vi)  NEW

3. NEW  BILL
   10  PRINT "BILL SMITH"
   20  END
   LIST
   RUN
```

```
5. 10  PRINT 1
   20  PRINT 2
   30  PRINT 3
   40  PRINT 4
   50  PRINT 5
   60  END
```

EXERCISE 1.3

```
1. (i)   56
         11
   (ii)  9   60
         STOP
   (iii) 10
         30
         20
   (iv)  THIS
         IS
         EASY
   (v)   5       10
         55      1
         THE     END

2. (i)   20  PRINT 7 + 5
   (ii)  30  PRINT "LOOK"
   (iii) 10  PRINT 8 * 9, 7 * 5
         20  PRINT 5 + 11, 6 + 9
   (iv)  30  PRINT 60 * 2; 50 * 2

3. 10  PRINT 12 + 25
   20  END

4. 10  PRINT 35 * 10
   20  END

5. 10  PRINT 120/3
   20  END

6. 10  PRINT 75 – 25
   20  END

7. 10  PRINT 20 ↑ 3
   20  END

8. 10  PRINT (15.6 + 7.52 + 3.90 + 6.57)/4
   20  END
```

EXERCISE 1.4

1. (1) 62
   (ii) 4596.4
   (iii) 0.057
   (iv) 629320
   (v) 0.00072254
   (vi) 248700
   (vii) 100005000
   (viii) 0.000175
   (ix) 0.00752
   (x) 2455700

2. (i) 0.56 E − 4
   (ii) 0.75 E + 6
   (iii) 0.1 E + 9
   (iv) 0.1 E − 5
   (v) 0.5675345 E + 9
   (vi) 0.5623 E + 2
   (vii) 0.5678 E − 4
   (viii) 0.301 E + 4
   (ix) 0.256 E + 6
   (x) 0.258001 E + 4

3. (i) 500 x 4000 = 2000000
   (ii) 0.02 x 50000 = 1000
   (iii) 640 ÷ 80 = 8
   (iv) 0.0056 ÷ 0.0007 = 8
   (v) 28000000000 + 1200000000 = 29200000000

4.
```
10  PRINT  1000 ↑ 4
20  END
```

5.
```
10  PRINT 0.6 E + 10 + 0.45 E + 9
20  END
```

6.
```
10  PRINT 9/5 E + 10
20  END
```

7.
```
10  PRINT 0.64 E + 5 * 0.5 E − 2
20  END
```

8.
```
10  PRINT 0.56 E + 8/0.7 E + 5
20  END
```

EXERCISE 1.5

1. (i) `30  LET C = 5`
   (ii) `30  LET F = D/E`
   (iii) `50  PRINT G, H, I, K`

2. (i)
```
50     5     250
```
   (ii)
```
7
```
   (iii)
```
100     15.5
```

3.
```
10  LET A = 15
20  LET B = 16
30  LET C = 25
40  LET D = A + B + C
50  PRINT D
60  END
```

4.
```
10  LET A = 6
20  PRINT A ↑ 2
30  LET A = 7
40  PRINT A ↑ 2
50  END
```

7.
```
10  LET A = 5
20  LET B = 10
30  LET C = 6
40  LET D = 6 * A * B + 3 * B * C
50  PRINT D
60  END
```

EXERCISE 1.6

1. (i) `20  LET B$ = "SMITH"`
   (ii)
```
20  LET D$ = "THERE!!"
30  PRINT C$, D$
```
   (iii) `50  PRINT A, B$, B, A$, A * B`

2. (i)
```
SAMPLE   PROGRAM
```
   (ii)
```
6.5   PLUS   3.5   =   10
```
   (iii)
```
IS   PROGRAMMING   EASY?
PROGRAMMING IS   EASY
```

3.
```
10  LET A$ = "PAUL"
20  LET B$ = "3 HARRY WAY"
30  LET C$ = "6155"
40  PRINT A$
50  PRINT B$, C$
60  END
```

6.
```
10  LET A$ = "HOW"
20  LET B$ = "YOU"
30  LET C$ = "DO"
40  PRINT  A$,  C$,  B$,  C$
50  END
```

EXERCISE 1.7

1. (i) 9
   (ii) 8
   (iii) 26
   (iv) 29
   (v) 31
   (vi) 80
   (vii) 126
   (viii) 300
   (ix) –74
   (x) 25

2. (i) 205
   (ii) 26
   (iii) 115
   (iv) 28
   (v) 360000
   (vi) 280
   (vii) 160000
   (viii) 2
   (ix) –100
   (x) 200000001

3. (i) 14
   (ii) 4
   (iii) 0.6666666
   (iv) 10
   (v) 6
   (vi) 162
   (vii) 1
   (viii) 12
   (ix) 0.24
   (x) 32.16666

4.
```
10  PRINT 10 * 5 + 4/2
20  END
```

7.
```
10  LET A = 5
20  LET B = 10
30  LET C = 3
40  LET D = 4
50  PRINT A + B * C – D
60  END
```

EXERCISE 1.8

1. (i) 5 * A + B
   (ii) 2 * P * R * T
   (iii) X * Y ↑ 2
   (iv) 3/4 * A * B
   (v) 2 * (A + B)
   (vi) 1/A + B
   (vii) 1/3 * L ↑ 2 * B
   (viii) (A + B) ↑ 3
   (ix) 2 * R * S + 3 * R ↑ 3
   (x) 1/2 (A ↑ 2 + B ↑ 3)

2. (i) 2 * P * R + 3 * S * T
   (ii) U ↑ 2 + 2 * A * S
   (iii) 1/(A + B)
   (iv) (B + C) / (2 * A)
   (v) U * T + 1/2 * A * T ↑ 2

3. (i) 1/2 * B * C ↑ 2 + A * D
   51
   (ii) (A * C) / (B – D)
   9
   (iii) A ↑ D + C * D ↑ D
   156
   (iv) A ↑ 2 + B ↑ 2 + A * B
   252
   (v) ((A – B + C) / C) ↑ 2
   9

4.
```
10  LET A = 5
20  LET B = 3
30  LET C = 4
40  LET D = 10
50  LET F = 2 * A * B + 3 * C * D
60  PRINT F
70  END
```

7.
```
10  LET R = 6
20  LET P1 = 3.14
30  LET C = 2 * P1 * R
40  PRINT "CIRCUMFERENCE IS", C
50  END
```

EXERCISE 1.9

1. (i) 9
   (ii) 10
   (iii) 29
   (iv) 2.75
   (v) 36
   (vi) 3.5
   (vii) 8
   (viii) 23
   (ix) –13
   (x) 26

2. (i) 7.5
   (ii) 6
   (iii) 1
   (iv) 8
   (v) 3

```
3. (i)   LET S = U * T + 1/2 * A * T ↑ 2
   (ii)  LET P = 1/2 * A * B ↑ 3 – 3 * D
   (iii) LET V = (U ↑ 2 + 2 * A * S) ↑ (1/2)
   (iv)  LET D = (X – Y) / (X ↑ 2 + Y ↑ 2)
   (v)   LET E = (X * Y + Y * Z) /(2 * X * Y)

4. 10  LET U = 10.54
   20  LET T = 5.63
   30  LET A = 2.54
   40  LET S = U * T + 1/2 * A * T ↑ 2
   50  PRINT S
   60  END

5. 10  LET A = 50.6
   20  LET B = 74.1
   30  LET C = 5.62
   40  LET D = (A + B + C) /3
   50  PRINT "AVERAGE IS" D
   60  END

6. 3    6    18
```

EXERCISE 1.10

```
1. 20  LET B = 20
   30  LET C = A * B
   50  END

2. THIS IS IT

3. A = 5

4. (i)   13
   (ii)  4
   (iii) 30
   (iv)  40

5. (i)   4 * X/3
   (ii)  8 * Y + 7
   (iii) X ↑ 2 + Y ↑ 2
   (iv)  (A + B) ↑ (1/2)
   (v)   4 * A * B * 5 * B * C
   (vi)  (R + 4) * (R – 3)
   (vii) A + (B + C) / 2
  (viii) (2 * X + 2 * Y) / (4 * R + 5 * T)
   (ix)  X ↑ 2 + 3 * X + 4
   (x)   6 * P / (5 * N ↑ 2)

6. 10  LET X = 11/4
   20  LET Y = 3 * X ↑ 3 – 5 * X ↑ 2 + 2 * X – 17
   30  PRINT "VALUE IS" Y
   40  END
```

```
7. 10  LET A = (15.7 + 30.42 + 160.3 + 49.7 + 16.4
                 + 1.79)/6
   20  PRINT "AVERAGE IS" A
   30  END

8. 10  LET A = 16.7
   20  LET Y = A ↑ 4
   30  PRINT Y
   40  END
```

EXERCISE 2.1

```
1. 10  PRINT "ENTER ANY NUMBER"
   20  INPUT C
   30  PRINT C "DOUBLED IS" C * 2
   40  END

2. 10  PRINT "ENTER 2 NUMBERS"
   20  INPUT A, B
   30  PRINT "THEIR DIFFERENCE IS" A – B
   40  END

3. 10  PRINT "ENTER 3 NUMBERS TO BE
                 ADDED"
   20  INPUT A, B, C
   30  PRINT "THE SUM IS" A + B + C
   40  END

5. 10  PRINT "ENTER THE TEMPERATURE IN
                 FAHRENHEIT"
   20  INPUT F
   30  LET C = 5 * (F – 32)   9
   40  PRINT C "DEGREES CELSIUS"
   50  END

10 10  PRINT "ENTER YOUR NAME, ADDRESS
                AND POSTCODE"
   20  INPUT A$, BS, CS
   30  PRINT C$, BS, AS
   40  END
```

EXERCISE 2.2

```
1. (i)  SAMPLE
   (ii) 5, 15, 20

2. (i)  UNCONDITIONAL
        LOOPS
        0
   (ii) 10, 15, 40, 50, 60, 70

3. (i)  1
        1
        1
        etc.
```

(ii) 10, 20, 30, 40, 10, 20, 30, 40, etc.

4. (i)
```
YOUR NAME
WHAT IS
```
(ii) 5, 6, 40, 50, 60, 10, 20, 30, 70

5. (i)

(ii) 10, 20, 30, 40, 50, 30, 40, 50, etc.

EXERCISE 2.3

1. (i) tells the user if a number input is positive or negative

(ii)
```
  6 IS A POSTIVE NUMBER
-51 IS A NEGATIVE NUMBER
  0 IS NOT POSITIVE NOR NEGATIVE
```

2.
```
  4
 14
 24
 34
 44



  .
 84
 94
104
```

3.
```
1
2
CAN YOU COUNT?
3
CAN YOU COUNT?
4
CAN YOU COUNT?
5
CAN YOU COUNT?
10
CAN YOU COUNT?
NO!!
```

6.
```
 5 LET A = 1
10 LET C = 2
20 LET A = A * C
30 LET C = C + 1
40 IF C < 21 THEN 20
50 PRINT "PRODUCT IS" A
60 END
```

EXERCISE 2.4

1.
```
10 PRINT "HOW MANY STARS ARE IN THE
             SOUTHERN CROSS?"
20 INPUT A
30 IF A = 5 THEN PRINT "CORRECT
                         ANSWER!"
40 IF A <> 5 THEN PRINT "WRONG, THE
   ANSWER IS 5"
50 END
```

2.
```
10 PRINT "ENTER 2 NUMBERS TO BE
   MULTIPLIED"
20 INPUT A, B
30 PRINT "PRODUCT IS" A * B
40 PRINT "DO YOU WANT ANOTHER GO?"
50 INPUT A$
60 IF A$ = "YES" THEN 10
70 PRINT "FINISHED"
80 END
```

6.
```
10 PRINT "ENTER 2 NUMBERS"
20 INPUT A, B
30 IF A > B THEN PRINT B, A
40 IF B > A THEN PRINT A, B
50 IF A = B THEN PRINT "SAME NUMBER"
60 END
```

EXERCISE 2.5

1. (i)
```
GOODBYE
GOODBYE
GOODBYE
GOODBYE
GOODBYE
GOODBYE
```
(ii)
```
NAMENAMENAMENAMENAME
```
(iii)
```
6    1
6    2
6    3
6    4
6    5
6    6
6    7
6    8
6    9
6   10
```
(iv)
```
FRED IS DEAD
FRED IS DEAD
FRED IS DEAD
FRED IS DEAD
FRED IS DEAD
```

```
(v)  22
     23
     24
     25
     26
     27
     28
     29
     30
     31
     32
     33
```

```
2. 10  FOR I = 1 TO 5
   20  PRINT "DOLLARS",
   30  NEXT I
   40  END
```

```
3. 10  FOR J = 1 TO 35
   20  PRINT "*";
   30  NEXT J
   40  END
```

```
6. 10  FOR 1 = 1 TO 11
   20  PRINT 2 * I - 1 ;
   30  NEXT I
   40  END
```

```
8. 10  FOR J = 1 TO 12
   20  PRINT J " x 5 = "J * 5
   30  NEXT J
   40  END
```

EXERCISE 2.6

```
1.  6
    8
   10
   12
   14
   16
   18
   20
```

```
2.  1    1
    2    4
    3    9
    4   16
    5   25
    6   36
    7   49
    8   64
    9   81
   10  100
```

3. 5

```
6. 10  PRINT "NUMBER", "SQUARE"
   20  FOR K = 40 TO 55
   30  PRINT K, K ↑ 2
   40  NEXT K
   50  END
```

EXERCISES 2.7

```
1. (i)    3 IS SQUARE ROOT OF 9
   (ii)   18
   (iii)  81     9        3
   (iv)   4     0.25     0.0625
```

```
2. 10  PRINT SQR (455)
   20  END
```

```
3. 10  PRINT "INPUT A NUMBER"
   20  INPUT X
   30  PRINT "SQUARE ROOT IS" SQR(X)
   40  END
```

```
5. 10  PRINT "NUMBER", "SQUARE", "SQUARE
                                     ROOT"
   20  FOR I = 1 TO 20
   30  PRINT I, I ↑ 2, SQR (I)
   40  NEXT I
   50  END
```

REVIEW EXERCISES (1) 2.8

```
1. (i)   THIS              (iv)   3
         IS                       4
         NOT                      5
         VERY                     6
         SENSIBLE                 7
  (ii)     1                      8
          16                      9
          49                     10
         100                     11
         STOP                    12
 (iii)   35                (v)    5
         20                      15
         3      3                10      35
```

```
2. (i)   I
         I
         I
         I
         I
  (ii)   3  8  13  18  23  28
 (iii)   25
         16
          9
          4
```

```
(iv)  2   1       (v)   1
      3   2
      3   2
      4   4
      4   3
      5   6
```

```
3. 10  FOR I = 1 TO 12
   20  PRINT 5 * I
   30  PRINT I
   40  END
```

```
7. 10  PRINT "INPUT KILOMETRES
                   TRAVELLED"
   20  INPUT K
   30  PRINT "INPUT ENGINE CAPACITY"
   40  INPUT E
   50  IF E < 1600 THEN 80
   60     LET A = 0.18 * K
   70     GOTO 90
   80  LET A = 0.15 * K
   90  PRINT "TRAVELLING ALLOWANCE IS
                         $" A
  100  END
```

REVIEW EXERCISES (2) 2.9

```
1. 10  PRINT "INPUT PRESENT HEIGHT"
   20  INPUT H
   30  FOR I = 1 TO 3
   40    H = H * 1.5
   50  NEXT I
   60  PRINT "FINAL HEIGHT IS" I
   70  END
```

```
2. 10  PRINT "INPUT TIME AS HOURS,
                MINUTES, PM/AM"
   20  INPUT H, M, T$
   30  IF T$ = "AM" THEN 50
   40    LET H = H + 12
   50  PRINT "TIME IS" H ":" M
   60  END
```

```
3. 10  PRINT "INPUT NUMBER OF SCORES"
   20  INPUT N
   30  FOR I = 1 TO N
   40    PRINT "INPUT NUMBER"
   50    INPUT M
   60  PRINT M;
   70    FOR J = 1 TO M
   80    PRINT "*" ;
   90    NEXT J
  100  PRINT
  110  NEXT I
  120  END
```

```
7. 10  PRINT "INPUT PRINCIPAL, INTEREST
              RATE, NUMBER OF YEARS"
   20  INPUT P, I, N
   30  FOR K = 1 TO N
   40  P = P + P * I
   50  NEXT K
   60  PRINT "FINAL AMOUNT IS" P
```

EXERCISE 3.1

1. The following message would appear on the screen.
   INPUT TWO NUMBERS WITH A COMMA BETWEEN
   ?
   If the user entered the numbers 5, 7 the message
   THE SUM IS 12
   would appear on the screen.

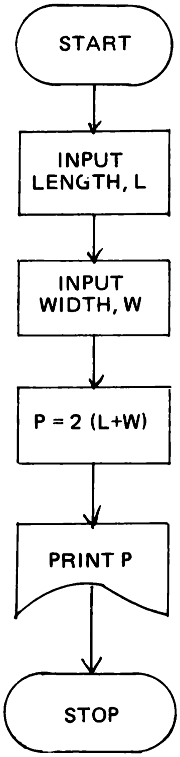
```
2. 10  REM A MULTIPLICATION PROGRAM :
   20  REM THIS PROGRAM MULTIPLIES TWO
                   NUMBERS
   30  REM PROGRAMMER : RON
   40  REM CREATED : DEC 1981
   50  PRINT "ENTER 2 NUMBERS SEPARATED
                  BY A COMMA"
   60  INPUT A, B
   70  PRINT "THE PRODUCT IS" A * B
   80  END
```

```
4. 10  REM THIS PROGRAM IS LIKE A
                    CALCULATOR:
   20  REM PROGRAMMER : RON
   30  REM CREATED : DEC 1981
   40  PRINT "THIS PROGRAM ADDS NUMBERS"
   50  PRINT "ENTER THE NUMBERS TO BE
                     ADDED"
   60  PRINT "SEPARATED BY PLUS SIGNS"
   70  PRINT "WHEN YOU WISH TO STOP"
   80  PRINT "ENTER THE = SIGN"
   90  PRINT "ENTER THE NUMBER TO BE
                     ADDED"
  100  INPUT A
  110  LET T = T + A
  120  REM T IS THE PROGRESSIVE TOTAL
  130  PRINT "ENTER A + SIGN TO ADD MORE
                     NUMBERS"
  140  PRINT "OR ENTER = TO OBTAIN THE
                     RESULT"
  150  INPUT A$
  160  IF A$ = "+" THEN 90
  170  REM MORE NUMBERS TO BE ADDED SO
                     BRANCH BACK
  180  IF A$ = "=" THEN 220
  190  REM A WRONG SYMBOL TRY AGAIN
  200  PRINT "YOU MUST ENTER + OR ="
```

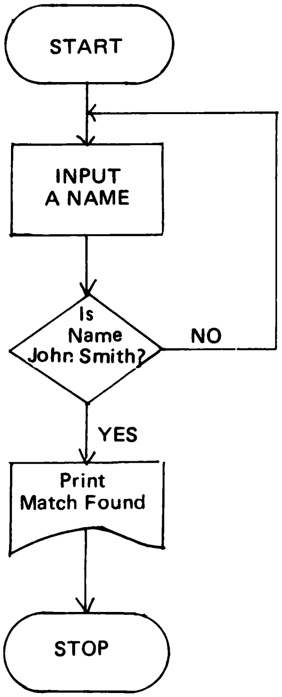
```
210  GO TO 130
220  PRINT "TOTAL IS" T
230  END
```

EXERCISE 3.2

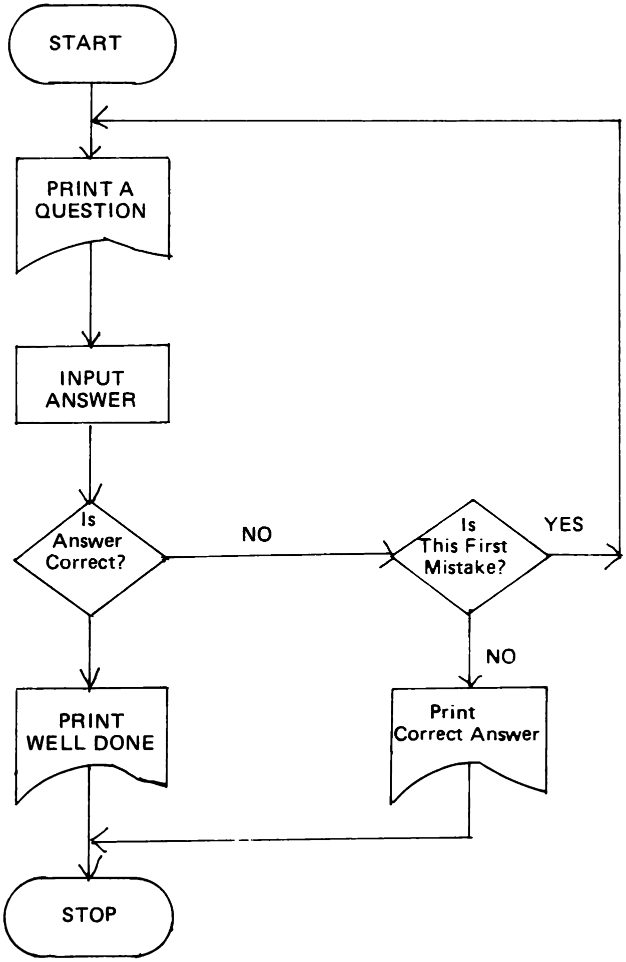
1.



2.



4.



EXERCISE 3.3

```
1. (i)     ------*
   (ii)    ---30
   (iii)   ----*------↑
   (iv)    ---4---8
   (v)     -*-*-*
   (vi)    ----COST---TOTAL
           ----++++---+++++
   (vii)   ---------0
           -----------0
           -------------0
           ---------------0
           -----------------0
           -------------------0

  (viii)   ---------------.
```

```
2. (i)    10  PRINT TAB (5) "*" ; TAB (8) "*"
   (ii)   10  PRINT TAB (5) "5"; TAB (7) "7";
                   TAB (9) "9"
   (iii)  10  PRINT TAB (10) "VALUE"; TAB (22)
                   "AMOUNT"
   (iv)   10  FOR I = 1 TO 4
          20    PRINT TAB (I + 4) " " "
   (v)    10  FOR I = 1 TO 7
          20    PRINT "*" ;
          30  NEXT I
          40  PRINT
          50  FOR I = 1 TO 3
          60     PRINT "*"; TAB (7) "*"
          70  NEXT I
          80  FOR I = 1 TO 7)
          90    PRINT "*" ;
         100  NEXT I
         110 END
```

```
7. 10  FOR I = 1 TO 40
   20    PRINT "*" ;
   30  NEXT I
   40  PRINT "*" ; TAB (40) "*"
   50  PRINT "*" ; TAB (40) "*"
   60  PRINT "* FRACTION"; TAB (20)
                                  "DECIMAL"
   70  PRINT "*"; TAB (40) "*"
   80  FOR I = 1 TO 8
   90  PRINT "*"; TAB(5)I; TAB (22) 1/8;
                    TAB (40) "*"
  100  NEXT I
  110  PRINT "*"; TAB (40) "*"
  120  FOR I = 1 TO 40
  130    PRINT "*" ;
  140  NEXT I
  150  END
```

EXERCISE 3.4

```
1. (i)  HELLO
        THIS PROGRAM PRINTS A FACTORIALS
                  TABLE
        1          1
        2          2
        3          6
        4         24
        5        120
        6        720
  (ii)  ERROR! TRY AGAIN – NUMBER MUST
                NOT BE NEGATIVE
        HELLO
        THIS PROGRAM PRINTS A FACTORIALS
                    TABLE
        1       1
        2       2
```

(iii) If a second negative number is input, the program doesn't trap the error.

```
2. 10  REM THIS PROGRAM CALCULATES THE
                   SQUARES
   12  REM OF THE WHOLE NUMBER 1–20 BUT
   14  REM ONLY PRINTS THE EVEN SQUARES
   16  REM AUTHOR : PAUL
   18  REM WRITTEN : NOV 1981
```

```
3. 10  PRINT "WHICH SHAPE DO YOU REQUIRE
                       THE AREA OF,"
   15  PRINT "TRIANGLE, CIRCLE OR
                        RECTANGLE?"
   20  INPUT A$
   30  IF A$ = "TRIANGLE" THEN GOSUB 70
   40  IF A$ = "CIRCLE" THEN GOSUB 110
   50  IF A$ = "RECTANGLE" THEN GOSUB 150
   60  STOP
   70  PRINT "ENTER THE HEIGHT AND
                        LENGTH OF BASE"
   80  INPUT H, B
   90  PRINT "THE AREA OF THIS TRIANGLE IS"
                        1/2 * B * H
  100  RETURN
  110  PRINT "ENTER THE RADIUS"
  120  INPUT R
  130  PRINT "THE AREA OF THIS CIRCLE IS"
                        3.14 * R ↑ 2
  140  RETURN
  150  PRINT "ENTER LENGTH AND WIDTH"
```

```
   160  INPUT L, W
   170  PRINT "THE AREA OF THIS RECTANGLE
                              IS" L * W
   180  RETURN
   190  END
```

```
4. 10  REM THIS PROGRAM CALCULATES
                          ESCAPE VELOCITIES,
   15  REM AND GRAPHS THE RESULTS
   20  REM AUTHOR : RON
   30  REM WRITTEN : DEC 1981
   40  DIM V(100)
   50  GOSUB 85
   60  GOSUB 180
   70  GOSUB 260
   80  STOP
   85  REM THIS SEGMENT CALCULATES
                          ESCAPE VELOCITIES
   90  PRINT "DISTANCE (M)", "ESCAPE
                          VELOCITY"
  100  FOR I = 1 TO 100
  110  LET R = I * 10000000
  120  LET G = 6.67 * 10E – 11
  130  LET M = 10E + 30
  140  LET V(I) = SQR (2 * G * M/R)
  150  PRINT  R, V(I)
  160  NEXT I
  170  RETURN
  180  REM THIS SEGMENT GRAPHS THE
                             RESULTS
  190  FOR I = 1 TO 100
  200  PRINT I * 100; TAB (V(I)/10 ↑ 6) "*"
  210  REM THE 10 ↑ 7 SCALES TO SCREEN SIZE
  220  NEXT I
  230  PRINT V(1); TAB (37) V (100)
  240  REM A SCALE FOR THE HORIZONTAL
                               AXIS
  250  RETURN
  260  LET V = 3 * 10E + 8
  270  LET R1 = 2 * G * M/V ↑ 2
  280  PRINT "DISTANCE FOR V OF 3 * 10E + 8
                              IS" R1
  290  RETURN
  300  END
```

EXERCISE 4.1

1. (i) 5
   (ii) 4
   (iii) –3
   (iv) 20
   (v) 2
   (vi) 9
   (vii) 6
   (viii) 2
   (ix) 9
   (x) 4

```
2. 10  PRINT "INPUT A NUMBER"
   20  INPUT N
   30  LET M = INT (N/7) * 7
   40  IF M <> N THEN 70
   50    PRINT "DIVISIBLE BY SEVEN"
   60    GOTO 80
   70  PRINT "NOT DIVISIBLE BY SEVEN"
   80  END
```

```
4. 10  PRINT "INPUT A NUMBER LESS THAN
                          100"
   20  INPUT N
   30  LET M = INT (N/2) * 2
   40  IF M = N THEN 70
   50    PRINT N "IS ODD"
   60    GOTO 80
   70  PRINT N "IS EVEN"
   80  END
```

```
6. 10  PRINT "INPUT AMOUNT OF MONEY"
   20  INPUT A
   30  LET I = A * 17/100
   40  LET J = INT (A * 100) / 100
   50  PRINT "17% OF THIS AMOUNT IS" J
   60  END
```

EXERCISE 4.2

1. (i) 5.24371
   (ii) 0
   (iii) 2
   (iv) 6
   (v) 25
   (vi) 0
   (vii) 0 to 4
   (viii) 1 to 6
   (ix) 0, 10, 20, ... 190
   (x) –4, –3, –2, –1, 0, 1, 2

```
2.  5  RANDOMIZE
   10  FOR I = 1 TO 10
   20    PRINT  RND
   30  NEXT I
   40  END
```

```
3.  5  RANDOMIZE
   10  PRINT  (RND * 10) + 1
   20  END
```

```
5.  5  RANDOMIZE
   10  LET N = INT (RND * 12) + 1
   20  LET M = INT (RND * 12) + 1
   30  PRINT N "X" M "=" ;
   40  INPUT A
   50  IF A = N * M THEN 80
   60    PRINT "INCORRECT ! ANSWER IS"
                          N * M
   70  GOTO 90
   80  PRINT "CORRECT ! WELL DONE"
   90  END
```

EXERCISE 4.3

```
1. 10  LET S = 0
   20  FOR I = 1 TO 20
   30    LET S = S + I
   40  NEXT I
   50  PRINT "SUM IS" S
   60  END
```

```
2. 10  LET S = 0
   20  FOR I = 1 TO 99 STEP 2
   30    LET S = S + I
   40  NEXT I
   50  PRINT "SUM IS" S
   60  END
```

```
3. 10  RANDOMIZE
   15  LET C = 0
   20  LET X = INT (RND * 100)
   30  LET Y = INT (RND * 100)
   40  LET Z = INT (RND * 100)
   50  PRINT X "+" Y "+" Z "=" ;
   60  INPUT A
   70  IF A = X + Y + Z THEN 110
   80    PRINT "INCORRECT. TRY AGAIN"
   90    LET C = C + 1
  100    GOTO 50
  110  END
```

```
5. 10  LET C = 0
   20  FOR I = 1 TO 20
   30  INPUT N
   40  IF N <> 7 THEN 60
   50  LET C = C + 1
   60  NEXT I
   70  PRINT "THERE WERE" C "SEVENS"
   80  END
```

EXERCISE 4.4

```
1. 10  RANDOMIZE
   20  LET H = 0
   30  LET T = 0
   40  FOR I = 1 TO 100
   50    LET N = INT (RND * 2)
   60    IF N = 0 THEN 90
   70      LET H = H + 1
   80      GOTO 100
   90    LET T = T + 1
  100  NEXT I
  110  PRINT "NUMBER OF HEADS:" H
  120  PRINT "NUMBER OF TAILS:" T
  130  END
```

```
2. 10  RANDOMIZE
   20  LET C1 = 0
   30  LET C2 = 0
   40  LET C3 = 0
   50  LET C4 = 0
   60  LET C5 = 0
   70  LET C6 = 0
   80  FOR I = 1 TO 600
   90    LET N = INT (RND * 6) + 1
  100    IF N <> 1 THEN
  110      LET C1 = C1 + 1
  120      GOTO 270
  130    IF N <> 2 THEN
  140      LET C2 = C2 + 1
  150      GOTO 270
  160    IF N <> 3 THEN
  170      LET C3 = C3 + 1
  180      GOTO 270
  190    IF N <> 4 THEN
  200      LET C4 = C4 + 1
  210      GOTO 270
  220    IF N <> 5 THEN
  230      LET C5 = C5 + 1
  140      GOTO 270
  250    IF N <> 6 THEN
  260      LET C6 = C6 + 1
  270  NEXT I
  280  PRINT "NUMBER OF ONES:" C1
  290  PRINT "NUMBER OF TWOS:" C2
  300  PRINT "NUMBER OF THREES:" C3
  310  PRINT "NUMBER OF FOURS:" C4
  320  PRINT "NUMBER OF FIVES:" C5
  330  PRINT "NUMBER OF SIXES:" C6
  340  END
```

[The ON GOTO statement simplifies this program greatly.]

```
5. 10  RANDOMIZE
   20  FOR I = 1 TO 25
   30    LET N = INT (RND * 365) + 1
   40    PRINT "DAY" N
   50  NEXT I
   60  END
```

[Can get program to check for repeated days using arrays -- introduced in the next chapter.]

In the simulating it is being assumed that each day is as likely to be a person's birthday as another.

REVIEW EXERCISES 4.5

```
1.  5 LET N = 0
   10  LET A = 0
   20  LET T = 0
   30  PRINT "INPUT TEMPERATURE"
   40  INPUT X
   50  LET N = N + 1
   60  LET A = A + X
```

```
 70 IF X <= 30 THEN 90
 80   LET T = T + 1
 90 PRINT "ANOTHER DAY? (Y/N) "
100 INPUT A$
110 IF A$ = "Y" THEN 30
120 PRINT "AVERAGE TEMPERATURE" A/N
130 PRINT "DAYS ABOVE 30 DEGREES" T
140 END
```

2.
```
10 RANDOMIZE
20 FOR I = 1 TO 20
30   LET N = INT (RND * 101)
40   PRINT N,
50 NEXT I
60 END
```

3. (i) 67
   (ii) –4
   (iii) 16
   (iv) 2

5. To ensure the program uses different random numbers on each run of the program.

6. (i) 6
   (ii) 1 or 2

PROJECT

```
 10 REM  BARBER SHOP SIMULATION
 20 REM
 30 REM THIS PROGRAM SIMULATES A
 40 REM BARBER SHOP WITH ONE SERVICE
 50 REM CHAIR AND FOUR WAITING CHAIRS.
 55 REM THE BARBER TAKES ABOUT
 60 REM 10 MINS PER CUSTOMER AND
 70 REM CUSTOMERS ARRIVE AT A RATE
 75 REM OF ABOUT 8 PER HOUR. THE
 80 REM SIMULATION RUNS FOR AN HOUR
 85 REM AND THE QUEUE LENGTH IS OUTPUT
 90 REM EVERY 5 MINUTES
 95 REM
100 REM * PROGRAMMER : NEWHOUSE DATE: 1982
101 REM
104 RANDOMIZE
105 REM * INITIALIZE COUNTERS
106 LET C = 0
107 LET T = 0
108 LET A = 0
109 LET N = 0
110 REM * SIMULATION FOR 60 MINUTES
112 FOR J = 1 TO 60
114   LET X = RND
116   IF X > 0.13 THEN 300
120   IF C = 4 THEN 280
125 REM * JOINS WAITING QUEUE
130   LET C = C + 1
140   GOTO 300
280 REM * CUSTOMER TURNED AWAY
290   LET A = A + 1
300   IF N = 0 THEN 380
305 REM * CUSTOMER ALREADY IN CHAIR
310     LET N = N + 1
320     IF N <> 10 THEN 400
325 REM * BARBER FINISHES A CUSTOMER
330       LET N = 0
340       GOTO 400
380     IF C = 0 THEN 400
385 REM * NEXT CUSTOMER TAKES SERVICE CHAIR
390     LET C = C – 1
395     LET N = 1
400   IF T <> 5 THEN 460
410 REM * PRINT QUEUE EVERY 5 MINUTES
415   LET T = 0
420   PRINT "QUEUE",
430   IF C = 0 THEN 457
435 REM * QUEUE HAS SOMEONE IN IT
440   FOR I = 1 TO C
450 PRINT "*  ";
455   NEXT I
457   PRINT
460   LET T = T + 1
470 NEXT J
480 PRINT
490 PRINT A "TURNED AWAY"
500 END
```

EXERCISE 5.1

1. (i)

C(1)	2
C(2)	4
C(3)	6
C(4)	8
C(5)	10

(ii)

A$(1)	MIKE
A$(2)	JOHN
A$(3)	FRED

(iii)

D(1)	1
D(2)	4
D(3)	9
D(4)	16
D(5)	25
D(6)	36

(iv)

A$(1)	BILL SMITH
A$(2)	21 FENTON ST.
A$(3)	MELVILLE

(v)

B(1)	1
B(2)	3
B(3)	6
B(4)	10
B(5)	15
B(6)	21

(vi)

A(1)	6	A$(1)	J + 1
A(2)	12	A$(2)	J + 1
A(3)	18	A$(3)	J + 1
A(4)	24	A$(4)	J + 1
A(5)	30	A$(5)	J + 1

(vii)

A$(1)	YES
A$(2)	YES
A$(3)	YES
A$(4)	YES
A$(5)	YES

(viii)

A(1)	1	A$(1)	B
A(2)	2	A$(2)	B
A(3)	3	A$(3)	B
A(4)	4	A$(4)	B
A(5)	5	A$(5)	B

(ix)

A(1)	1	B(1)	2	C(1)	3
A(2)	2	B(2)	4	C(2)	6
A(3)	3	B(3)	6	C(3)	9
A(4)	4	B(4)	8	C(4)	12
A(5)	5	B(5)	10	C(5)	15
A(6)	6	B(6)	12	C(6)	18
A(7)	7	B(7)	14	C(7)	21
A(8)	8	B(8)	16	C(8)	24
A(9)	9	B(9)	18	C(9)	27
A(10)	10	B(10)	20	C(10)	30

EXERCISE 5.2

1. (i)

```
 10
 20
 30
 40
 50
 60
 70
 80
 90
100
```

(ii)

```
HELLO
HELLO
HELLO
HELLO
HELLO
```

(iii)

```
11    1
12    2
13    3
14    4
15    5
```

(iv)

```
*     3
*     2.5
*     2
*     1.5
*     1
*     0.5
```

(v)

```
24
28
32
36
40
```

(vi)

```
6   24   54   96   150
```

2.

```
10  DIM A(20)
20  FOR I = 1 TO 20
30    A(I) = I
40  NEXT I
50  FOR I = 1 TO 20
60    PRINT A(I)
70  NEXT I
80  END
```

3.

```
10  DIM B(50)
20  FOR I = 1 TO 50
30    B(I) = 2 * I – 1
40  NEXT I
50  FOR I = 1 TO 50
60    PRINT B(I)
70  NEXT I
80  END
```

6.

```
10  DIM A$(20), A(20)
20  PRINT "INPUT NAME, AGE : 20 TIMES"
30  FOR I = 1 TO 20
40    INPUT A$(I), A(I)
50  50 NEXT I
60  END
```

9.

```
10  DIM B(20)
20  PRINT "INPUT 20 SCORES – ONE PER LINE"
30  FOR I = 1 TO 20
40    INPUT B(I)
50    S = S + B(I)
60  NEXT I
70  PRINT "AVERAGE IS" S/20
80  END
```

10.

```
10  DIM N(30)
20  PRINT "INPUT NUMBER OF SCORES"
30  INPUT N
40  PRINT "INPUT SCORES – ONE PER LINE"
```

```
 50  FOR I = 1 TO N
 60    INPUT N(I)
 70  NEXT I
 80  PRINT "INPUT PASS MARK"
 85  INPUT P
 90  FOR I = 1 TO N
100    IF N(I) < P THEN 120
110    C = C + 1
120  NEXT I
130  PRINT C "PASSED"; TAB(15) N-C
                        "FAILED"
140  END
```

EXERCISE 5.3

```
1. (i)  7    6    5        (v) 1      ONE
  (ii)  HELLO                  2      TWO
 (iii)  +                      3      THREE
        4     6.2              4      FOUR
        *                      5      FIVE
        FINISH
  (iv)  -1
        -2
        -3
```

```
2. 10  READ A, B, C
   20  LET D = A + B + C
   30  PRINT "SUM IS" D
   40  DATA 107.5, 623.5, 575
   50  END
```

```
7. 10  READ A, R, N
   20  LET T = A * R ↑ N
   30  LET S = A * (1-R) ↑ N/(1-R)
   40  PRINT "T = " T ; TAB(15) "S = " S
   50  DATA 12, 0.75, 5
   60  END
```

```
8. 10  DIM A$(5), B$(5), C(5)
   20  LET N = 0
   30  FOR I = 1 TO 5
   40    READ A$(I), B$(I), C(I)
   50  NEXT I
   60  FOR I = 1 TO 5
   70    PRINT
   80    PRINT B$(I); " " A$(I), C(I)
   90    N = N + C(I)
  100  NEXT I
  110  PRINT "AVERAGE AGE IS" N/5
  120  DATA "SMITH", "BILL", 22
  121  DATA "LISPET", "JANE", 51
  122  DATA "DEAN", "JAMES", 49
  123  DATA "TAYLOR", "JAMES", 35
  124  DATA "EVERT", "CHRISTINE", 24
  130  END
```

```
10. 10  DIM N$(6), P(6), N(6)
    20  LET C = 0
    25  LET C1 = 0
    30  FOR I = 1 TO 6
    40    READ N$(I), P(I), N(I)
    50  NEXT I
    60  FOR I = 1 TO 6
    70    LET C = C + N(I)
    80    LET C1 = C1 + N(I) * P(I)
    90  NEXT I
   100  PRINT "TOTAL NUMBER OF CARS" C
   110  PRINT "MARKET VALUE" C1
   120  DATA "SEDAN", 5600, 15
   121  DATA "UTILITY", 7100, 5
   122  DATA "STATION WAGON", 6950, 9
   123  DATA "SPORTS", 6500, 3
   124  DATA "PANEL VAN", 7250, 2
   125  DATA "TRUCK", 11050, 1
   130  END
```

EXERCISE 5.4

```
1. (i)   10  DIM B(4, 5)
  (ii)   a)  7
         b) 11
         c)  4
         d) 13
 (iii)   a) B(2, 1)
         b) B(3, 5)
         c) B(1, 4)
         d) B(4, 5)
  (iv)   10  FOR I = 1 TO 4
         20    FOR J = 1 TO 5
         30      PRINT B(I, J),
         40    NEXT J
         50    PRINT
         60  NEXT I
```

```
2. (i)   10  DIM B$(5, 3)
  (ii)   SEDAN
         1976
         1979
 (iii)   1973
         1976
         1980
         1979
         1977
```

(iv)

HOLDEN	SEDAN	1973
FORD	UTE	1976
TOYOTA	LIFTBACK	1980
SUBARU	SEDAN	1975
SIGMA	WAGON	1977

3. (i) 10 DIM A(4, 5), A$(6, 2)

(ii) a) A(2, 2)
  b) A(3, 4)
  c) A$(4, 1)
  d) A$(4, 2)

(iii) a) 11
  b) B
  c) 17
  d) D

(iv)
```
a) 19        b) A        c) 20  A
   15           C           15  B
   11           E
    7           G
    3           I
                K
```

(v)
```
a) 100  FOR I = 1 TO 5
   110    PRINT A(3, I),
   120  NEXT I
b) 100  FOR I = 1 TO 6
   110    PRINT A$(1, 2)
   120  NEXT I
c) 100  FOR I = 1 TO 4
   110    PRINT A(I, 5)
   120  NEXT I
```

4. (i) A$

JAMES	JOHN
HARRY	BILL
FRED	MIKE

(ii) A

180	200
220	240
260	280

C$

ONE	TWO
THREE	FOUR
FIVE	SIX

5.
```
  5  DIM D(5, 6)
 10  FOR I = 1 TO 5
 20    FOR J = 1 TO 6
 30      LET D(I, J) = 5
 40    NEXT J
 50  NEXT I
 60  REM OUTPUT ARRAY
 70  FOR I = 1 TO 5
 80      FOR J = 1 TO 6
 90       PRINT D(I, J),
100      NEXT J
110      PRINT
120  NEXT J
130  END
```

8.
```
 10  DIM N$(6), S(6, 4)
 20  PRINT "INPUT NAME, MARK 1, MK 2,
              MK 3, MK 4"
 30  FOR I = 1 TO 6
 40    INPUT N$(I), S(I, 1), S(I, 2), S(I, 3), S(I, 4)
 50  NEXT I
 60  PRINT "NAME", "MARK 1", "MARK 2",
                "MARK 3", "MARK 4"
 70  FOR I = 1 TO 6
 80    PRINT N$(I),
 90    FOR J = 1 TO 4
100      PRINT S(I, J),
110    NEXT J
120    PRINT
130  NEXT I
140  END
```

EXERCISE 5.5

1. line 100 – 5
   line 150 – 15

2.
```
 10  DIM N(10)
 20    PRINT "INPUT 10 NUMBERS – ONE PER
                              LINE"
 30  FOR I = 1 TO 10
 40    INPUT N(I)
 50  NEXT I
 60  LET S = N(1)
 70  FOR I = 2 TO 10
 80    IF N(I) > S THEN 100
 90      LET S = N(I)
100  NEXT I
110  PRINT "SMALLEST IS" S
120  END
```

3.
```
 10  DIM A$(100)
 20  PRINT "INPUT NUMBER OF NAMES
                   (< 100)"
 30  INPUT N
 40  PRINT "INPUT NAMES – ONE PER LINE"
 50  FOR I = 1 TO N
 60  INPUT A$(I)
 70  NEXT I
 80  LET F$ = A$(1)
 90  LET L$ = A$(1)
100  FOR I = 2 TO N
110    IF A$(I) > F$ THEN 130
120      LET F$ = A$(I)
130    IF A$(I) < L$ THEN 150
140      LET L$ = A$(I)
150  NEXT I
160  PRINT "FIRST NAME IS" F$
170  PRINT "LAST NAME IS" L$
180  END
```

EXERCISE 5.6

```
1. (i) 200  FOR I = 1 TO 3
       210    FOR J = I + 1 TO 4
       220      IF A(I) > A(J) THEN 260
       230        T = A(I)
       240        A(I) = A(J)
       250        A(J) = T
       260    NEXT J
       270  NEXT I
```

(ii)
and
(iii)

6	10	10	10	10	10	10	A(1)
10	6	6	6	6	8	8	A(2)
5	5	5	5	5	5	6	A(3)
8	8	8	8	8	6	5	A(4)
1	1	1	2	2	3		I
2	3	4	3	4	4		J

```
2.  10  DIM A(20)
    20  PRINT "INPUT NUMBERS"
    30  FOR I = 1 TO 20
    40    INPUT A(I)
    50  NEXT I
    60  FOR I = 1 TO 19
    70    FOR J = I + 1 TO 20
    80      IF A(I) > A(J) THEN 120
    90        T = A(I)
   100        A(I) = A(J)
   110        A(J) = T
   120    NEXT J
   130  NEXT I
   140  REM OUTPUT SORTED LIST
   150  FOR I = 1 TO 20
   160    PRINT A(I)
   170  NEXT I
   180  END
```

3. Same as question 2 except use A$ in place of A, T$ in place of T and 25 in place of 20.

EXERCISE 6.1

1.

BILL	439	24		PAUL	621	15	
SMITH	2111	12	4	KING	3011	11	10
		81				81	
Name	Tel.	Date Admtd.	Doctor				

2. Fields e.g.
article number
article name
wholesale price
retail price
number in stock
supplier number
minimum number

3. e.g.

Book Title	Author Name	Dewey Classification	Out on Loan (Yes/No)	No. of Cppies

EXERCISE 6.2

```
1. 10  FILES NAMEFL
   20  PRINT "INPUT NAMES – ONE PER LINE"
   30  PRINT "TYPE ZZZ TO FINISH"
   40  INPUT N$
   50  IF N$ = "ZZZ" THEN 80
   60  PRINT #1, N$
   70  GOTO 40
   80  END
```

```
2. 10  FILES NAMEFL
   20  INPUT #1, NS
   30  PRINT N$
   40  IF END #1, THEN 60
   50  GOTO 20
   60  END
```

```
3.  5  PRINT "NAME", "PAY SCALE"
   10  FILES EMPLOY
   20  INPUT #1, NS, P
   30  PRINT N$, P
   40  IF END #1, THEN END
   50  GOTO 20
   60  END
```

4. To write back to the same file, after reading all records into arrays, SCRATCH the file and write the arrays back into it.

EXERCISE 7.1

```
1. 10  PRINT "THIS PROGRAM FINDS
                    ABSOLUTE VALUES"
   20  PRINT "ENTER ANY NUMBER"
   30  INPUT A
   40  PRINT ABS(A)
   50  END
```

```
2.  10 REM THIS PROGRAM FINDS ROOTS OF
                  QUADRATIC EQUATIONS
    20 REM PROGRAMMER : RON
    30 REM CREATED : JAN 1982
    40 REM * * * * * * * * *
    50 PRINT "INPUT THE COEFFICIENTS OF
                 YOUR QUADRATIC"
    60 PRINT "INPUT COEFFICIENT OF X ↑ 2"
    70 INPUT A
    80 PRINT "INPUT COEFFICIENT OF X"
    90 INPUT B
   100 PRINT "INPUT CONSTANT"
   110 INPUT C
   120 REM CHECK EQUATION HAS REAL
                   ROOTS
   130 IF B ↑ 2 – 4 * A * C > = 0 THEN 160
   140 PRINT "THIS EQUATION HAS NO REAL
               ROOTS, "ENTER NO'S AGAIN"
   150 GO TO 50
   160 LET D = (–B + SQR (B ↑ 2 – 4 * A * C) )/
                   (2 * A)
   170 LET E = (–B – SQR (B ↑ 2 – 4 * A * C) )/
                   (2 * A)
   180 PRINT "ROOTS ARE" D, E
   190 REM MORE EQUATIONS?
   200 PRINT "DO YOU WISH TO TRY AGAIN"
   210 INPUT A$
   220 IF A$ = "YES" THEN 50
   230 PRINT "PROCESSING COMPLETE"
   240 END
```

```
3.  10 PRINT "ENTER NO OF RADIANS"
    20 INPUT A
    30 PRINT "SIN ("A") = " SIN (A)
    40 END
```

```
5.  10 FOR D = 1–180 to 180 STEP 10
    20 LET R = D/180 * 3.1416
    30 REM CONVERTS DEGREES TO RADIANS
    40 LET S = SIN(R)
    50 REM PRINT DEGREES AND GRAPH SIN
    60 PRINT D;  TAB (S * 15 + 20) ; "*"
    70 REM S * 15 SCALES UP AND + 20 MOVES
                 GRAPH ACROSS
    80 NEXT D
    90 REM SCALE ACROSS BOTTOM
   100 LET A = 5
   110 FOR I = –1 TO 1 STEP 0.5
   120 PRINT TAB(A) ; I;
   130 LET A = A + 7
   140 NEXT I
   150 END
```

EXERCISE 7.2

1. (i) 8
   (ii) –1
   (iii) 6
   (iv) –28

```
2. (i)   DEF FNA(X) = X ↑ 2 + 3
   (ii)  DEF FNB(Y) = Y ↑ 3 + 2 * Y ↑ 2 + 3 * Y + 5
   (iii) DEF FNH(X) = SIN(X) ↑ 2 + COS(X) ↑ 2
   (iv)  DEF FNF(X, Y) = X ↑ 2 + Y ↑ 2
   (v)   DEF FNG(X, Y) = X/Y + 2 * X * Y
```

```
3.  10 DEF FNF(C) = 9/5 * C + 32
    20 PRINT FNF(10)
    30 END
```

```
5.  10 PRINT "ENTER %"
    20 INPUT P
    30 PRINT "ENTER AMOUNT"
    40 INPUT A
    50 DEF FNP(X, Y) = X/100 * Y
    60 PRINT FNP(P, A)
    70 END
```

EXERCISE 7.3

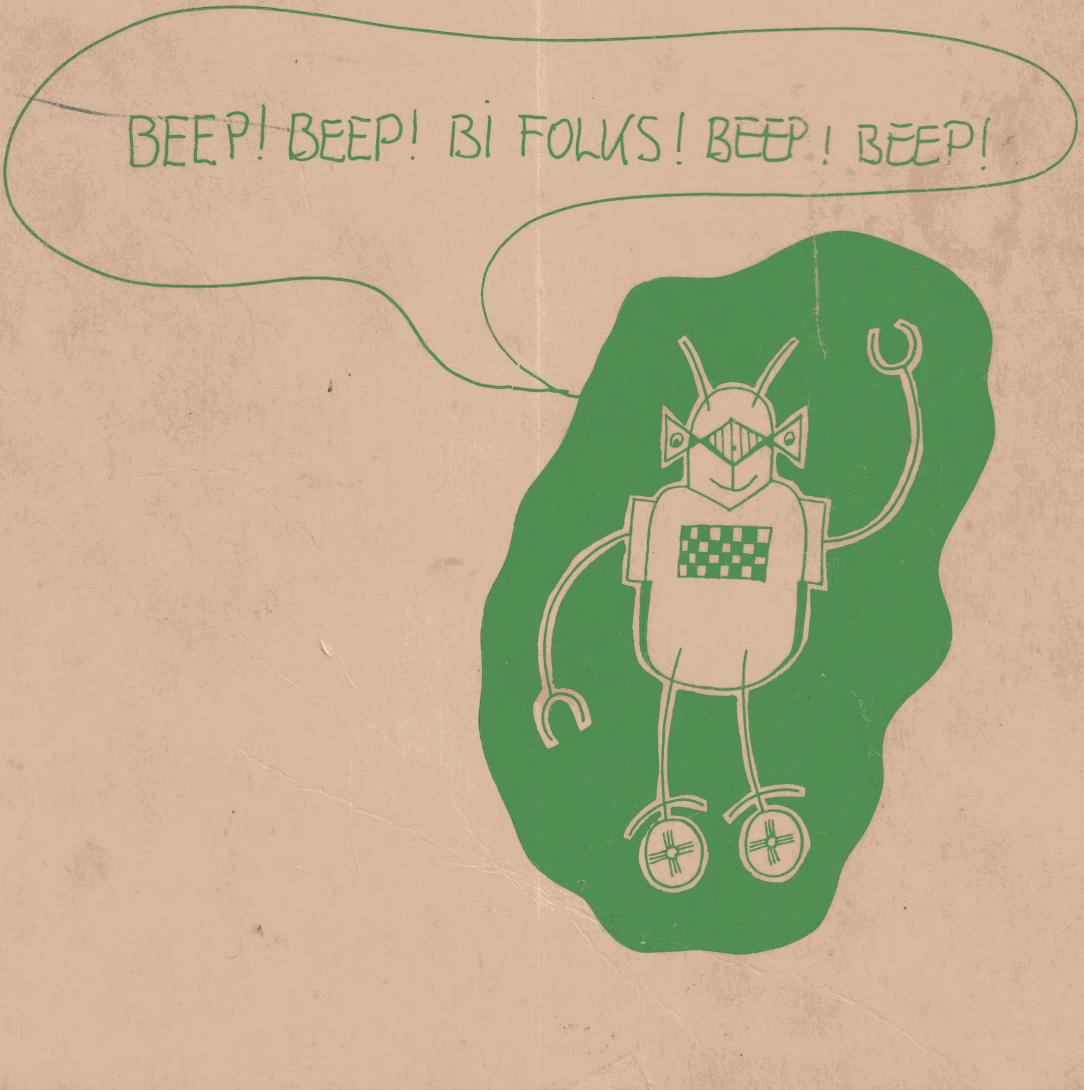
1. (i) 1, 2, 3 and 4
   (ii) line 120
   (iii) 2
   (iv) 120 ON A GOTO 100, 160, 120, 140, 200

2. (i) 3
   (ii) 6
   (iii) 4
   (iv) line 230

```
3.  40 PRINT "INPUT THE RADIUS"
    50 INPUT R
    60 PRINT "INPUT 1 FOR AREA OR 2 FOR
                  CIRCUMFERENCE"
    70 INPUT X
    80 ON X GOTO 100, 200
   100 REM AREA OF A CIRCLE
   110 LET A = 3.14 * R ↑ 2
   120 PRINT "AREA OF CIRCLE:" A
   130 GOTO 240
   200 REM CIRCUMFERENCE OF A CIRCLE
   210 LET C = 2 * 3.14 * R
   220 PRINT "CIRCUMFERENCE OF CIRCLE:" C
   240 END
```

```
6.  30 DIM A(100)
    40 PRINT "INPUT NUMBER OF MARKS TO
                     ENTER"
```

```
 50 INPUT N
 60 PRINT "INPUT MARKS - ONE PER LINE."
 70 FOR I = 1 TO N
 80   INPUT A(I)
 90 NEXT I
100 PRINT
110 PRINT "WHICH OF THE FOLLOWING DO
                     YOU REQUIRE?"
120 PRINT "1 - THE AVERAGE MARK"
130 PRINT "2 - THE HIGHEST MARK"
140 PRINT "3 - THE LOWEST MARK"
150 PRINT "4 - THE RANGE OF MARKS"
160 PRINT "INPUT A NUMBER."
170 INPUT X
180 ON X GOTO 200, 300, 400, 500
200 REM FIND AVERAGE MARK
210 FOR I = 1 TO N
220   LET S = S + A(I)
230 NEXT I
240 LET A = S/N
250 PRINT "AVERAGE MARK IS:" A
260 GOTO 600
300 REM FIND HIGHEST MARK
310 FOR I = 1 TO N
320   IF H > A(I) THEN 340
330   LET H = A(I)
340 NEXT I
350 PRINT "HIGHEST MARK:" H
360 GOTO 600
400 REM FIND LOWEST MARK
410 LET L = A(1)
411 FOR I = 2 TO N
420   IF L < A(I) THEN 440
430   LET L = A(I)
440 NEXT I
450 PRINT "LOWEST MARK:" L
460 GOTO 600
500 REM FIND THE RANGE OF MARKS
510 LET L = A(I)
520 FOR I = 1 TO N
530   IF H > A(I) THEN 550
540   LET H = A(I)
550   IF L < A(I) THEN 570
560   LET L = A(I)
570 NEXT I
580 LET R = H-L
590 PRINT "RANGE OF MARKS:" R
600 PRINT "ANOTHER GO? (Y/N)"
620 IF Q$ = "Y" THEN 110
630 END
```


BEEP! BEEP! Bi FOLKS! BEEP! BEEP!